

TroyMaster

v1.0.3

User Manual

A TABLE OF CONTENTS

| | | |
|------|-----------------------------------|----|
| A | Table of Contents | 2 |
| B | Preface | 4 |
| B1 | For Users | 4 |
| B2 | For Administrators | 4 |
| B3 | For Developers | 4 |
| C | What is TrayMaster? | 5 |
| C1 | Warehouse Modelling | 6 |
| C2 | Why this app? | 7 |
| D | How to use TrayMaster | 9 |
| D1 | Accessing TrayMaster | 9 |
| D1.1 | Device Requirements | 9 |
| D1.2 | Installing TrayMaster | 9 |
| D1.3 | Versioning | 9 |
| D2 | Loading, Errors and Dialogs | 10 |
| D3 | Sign in | 13 |
| D3.1 | Sign in Area | 13 |
| D4 | Main Menu | 14 |
| D5 | Switch Warehouse | 15 |
| D6 | Shelf View | 16 |
| D6.1 | Side Panel | 17 |
| D6.2 | Visualisation | 18 |
| D6.3 | Selection | 19 |
| D6.4 | Input Keyboards | 22 |
| D6.5 | Auto-advance | 26 |
| D6.6 | Tray Information Dialog | 27 |
| D6.7 | Edit Shelf | 28 |
| D6.8 | Navigation | 29 |
| D7 | Find | 31 |
| D8 | Settings | 32 |
| D8.1 | Personal Settings | 32 |
| D8.2 | Category Editor | 34 |
| D8.3 | Zone Editor | 35 |

| | | |
|------|--------------------------------------|----|
| E | Troubleshooting..... | 37 |
| F | Maintenance and Administration | 39 |
| F1 | Firebase Fees..... | 39 |
| F2 | Firebase Console | 40 |
| F2.1 | Warehouse and User Management | 40 |
| G | Developer Guide | 43 |
| G1 | Introduction | 43 |
| G2 | Technologies Used | 43 |
| G3 | Code Repository..... | 43 |
| G4 | Continuous Integration..... | 44 |
| G5 | Continuous Deployment (Heroku)..... | 44 |
| G6 | Firebase..... | 45 |
| G6.1 | Setup | 45 |
| G6.2 | Authentication | 46 |
| G6.3 | Database..... | 46 |
| G6.4 | Functions | 46 |
| G6.5 | Hosting | 46 |
| G7 | Warehouse Model | 47 |
| G8 | React..... | 48 |

B PREFACE

This preface aims to guide you towards the sections of the manual relevant to you and to give you the guidance and information necessary to read them.

B1 FOR USERS

If you want to find out whether your device supports TrayMaster, read *D1 Accessing TrayMaster*.

If you are unfamiliar with the food store process or if you want to familiarise yourself with the warehouse model used by TrayMaster, read *C What is TrayMaster?*.

If you want to discover/understand all features of the system, read *D How to use TrayMaster*.

If you're troubleshooting an issue regarding a feature of the system, read *E Troubleshooting*.

B2 FOR ADMINISTRATORS

If you want to edit the layout of the warehouse, read *D8.3 Zone Editor*.

If you want to know the possible fees for upkeep of the system, read *F1 Firebase Fees*.

If you want to manage the existing users, read *F2.1 Warehouse and User Management*.

If you want to set up a new warehouse, read *F2.1 Warehouse and User Management*.

B3 FOR DEVELOPERS

If you want to gain access to the TrayMaster source code, read *G3 Code Repository*.

If you want to get an idea of the prerequisite knowledge needed to edit the source code, read *G2 Technologies Used*.

If you want to know how to fully redeploy TrayMaster, read *G5 Continuous Deployment (Heroku)*.

If you want to set up a new Database, read *G6 Firebase*.

C WHAT IS TRAYMASTER?

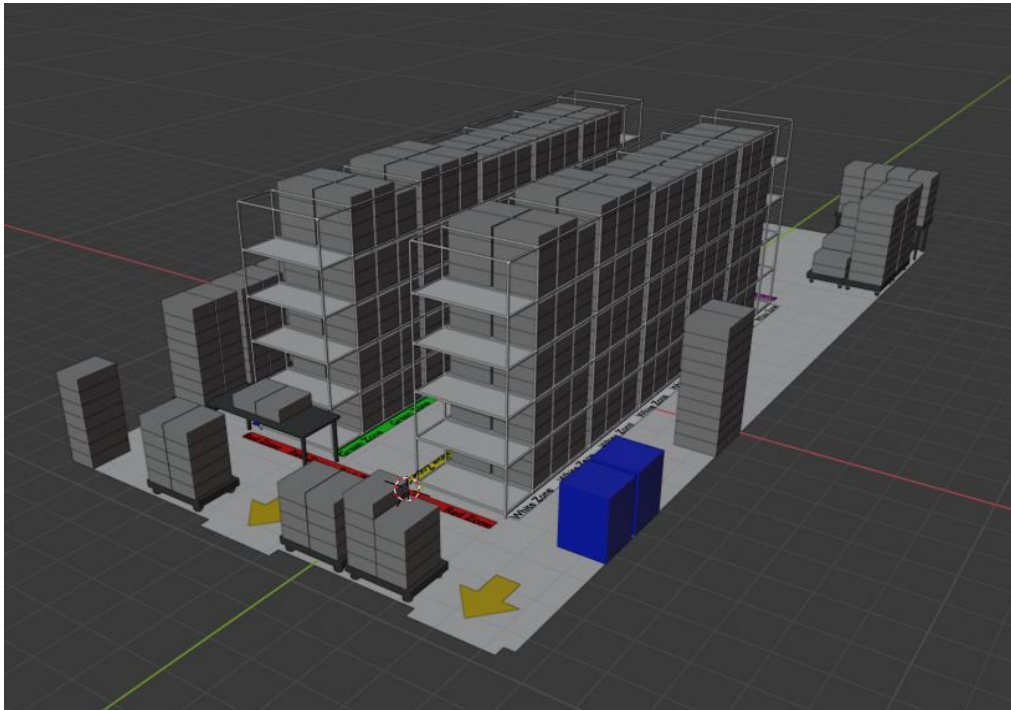


FIGURE 1. 3D RENDERING OF A FOODBANK WAREHOUSE IN CHESTER-LE-STREET.

TrayMaster is a software application primarily intended for managing trays of perishable stock on shelves inside a warehouse. It has been designed following consultation and analysis of the processes currently in use at Durham Christian Partnership's Chester-le-Street foodbank (see *Figure 1*); but more generally will be applicable to many other Trussell Trust-affiliated foodbanks across the UK, as well as other sectors with physical stock located within trays where incoming stock is hard to predict. This software should complement existing workflows and aid warehouse staff in their day-to-day activities.

- A warehouse can be any building that stores stock in trays on shelves. A tray is atomic (homogeneous, containing one type of item) and thus is treated by the application as the smallest object to record. For example, the app can record that in the second column, bottom row of shelf A₃ in the Green Zone, there is a tray of BEANS.

Processing often begins when a warehouse receives new stock (specifically donations in a foodbank). The application is optimised for use cases such as these since it relies on no prior knowledge of incoming stock. Staff can simply use this app to perform a stocktake and record where they place new trays of stock/unexpected donations, as soon as they arrive.

Clearly when dealing with perishable items, it is imperative to record expiry dates to minimise wastage and enable users to identify soon-to-expire stock that should be dispatched promptly. This app gives users the ability to record the expiry date of any tray, with custom precision - for some items, the expiry needn't be known any more accurately than the year. But for some items, the expiry date must be recorded down to the quarter or even month.

As such, this app has been built with flexibility in mind: a user can record tray expiry with as much precision as desired, optionally record the weight of trays, and even leave a custom text comment on a tray to record other important data.

C1 WAREHOUSE MODELLING

Users should carefully consider how their warehouse could be modelled using TrayMaster. Firstly, it should be split into "zones", which are the largest possible divisions of a warehouse. Zones provide for easy navigation and spatial intuition when navigating in the app.

If some storage doesn't naturally fit under a designated zone, users can create a custom zone for these locations. *Figure 2* shows one such hypothetical zone.

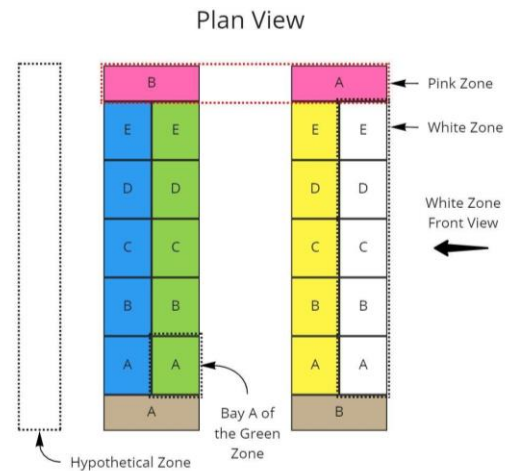


FIGURE 2. PLAN VIEW OF AN EXAMPLE WAREHOUSE.

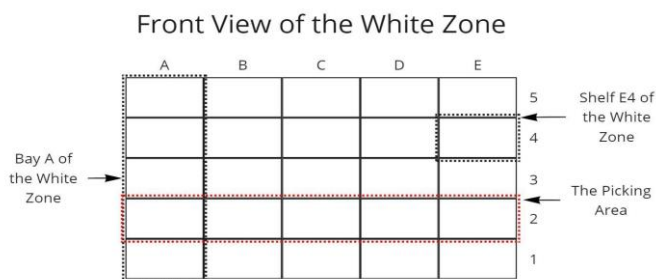


FIGURE 3. FRONT VIEW OF MULTIPLE BAYS WITHIN A ZONE.

Figure 3 shows the front view of a zone. One zone usually represents the entire side of a shelving rack. Such a rack is further divided along its length at the joints into vertical "bays". Each cell in this figure is a single shelf. The "picking area" is the row of shelves that is accessible by hand for pickers to put into orders – generally the row of shelves second closest to the ground (around waist height).

Figure 4 shows a shelf. A shelf contains columns, which are stacks of trays. The shelf here has four columns, each containing three trays.

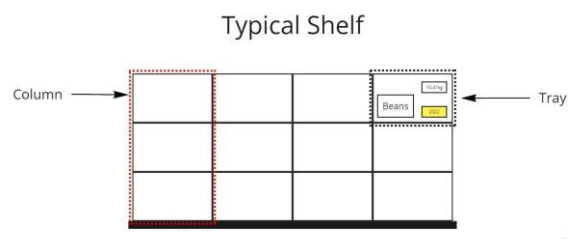


FIGURE 4. FRONT VIEW OF A STANDARD SHELF.

This application has been built with configurability in mind, to ensure that it remains applicable under many use cases. Users can model warehouses with custom sizes of zones, bays, shelves and columns. This manual will explore how to configure the dimensions and size of each of these.

Furthermore, virtually any warehouse can be represented using the app's generic model. Consider "shelf-like storage" such as a floor dolly (see Figure 5). Within TrayMaster, this could be considered as a shelf with two columns and a maximum height of five trays. Then, multiple dollies can come together to form a zone (see Figure 6). Using this idea, many different storage areas can be represented as shelf-like objects.

Shelf-Like (Dolly)

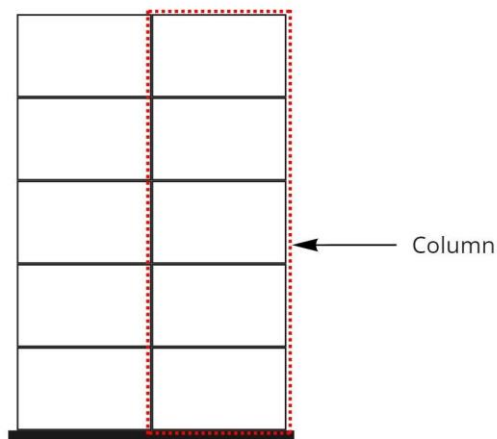


FIGURE 5. FRONT VIEW OF A DOLLY

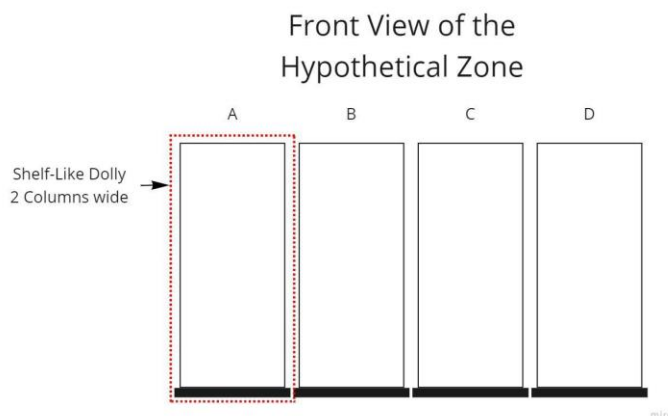


FIGURE 6. FRONT VIEW OF THE HYPOTHETICAL ZONE, CONTAINING DOLLIES.

C2 WHY THIS APP?

Usually, a primary aim for warehouse staff is to minimise their stock wastage. TrayMaster provides users with better insights into the current state of their warehouse to enable better-informed decisions on dispatching items which will expire soon. In some cases, it will even allow users to identify the categories of item in which they are understocked – justifying specific food donation drives for foodbanks, or new orders for specific stock from other warehouses.

TrayMaster has also been built with speed in mind, since warehouse staff will only maintain data accuracy if it remains simple and quick to update the app.

D HOW TO USE TRAYMASTER

This section will explore how to access and use the TrayMaster app.

D1 ACCESSING TRAYMASTER

TrayMaster can be accessed at <https://traymaster.herokuapp.com>

When hosted on the free Heroku plan, the app enters hibernation mode after 30 minutes. It may take around 20 seconds longer to load if it hasn't been visited for a while.

D1.1 DEVICE REQUIREMENTS

Browser: Chromium-based (Google Chrome, new Microsoft Edge); or
Mozilla Firefox

Minimum Display Size: 7"

Minimum Resolution*: 800x480

Input method: Touchscreen; or
Mouse and keyboard

*This is rendering ('viewport') resolution, calculated by $\frac{\text{display resolution}}{\text{display scaling}}$.

Specifications outside the above may function but have no guaranteed support.

D1.2 INSTALLING TRAYMASTER

TrayMaster can be installed as a Progressive Web App (PWA). A PWA is a website that can be installed onto your device, like any other app or program. Installing TrayMaster as a PWA allows the app to cache some information on your device, ensuring faster loading times, as well as some platform-specific improvements such as displaying in full screen. One disadvantage however is that you may occasionally need to delete and reinstall the app to guarantee that you are using the latest version.

Your browser will usually prompt you to **install the TrayMaster PWA** when you first visit. If not, you can search online for a guide to installing a PWA that is relevant to your browser and operating system (usually an 'Add to Home Screen' button or similar).

D1.3 VERSIONING

The TrayMaster version number can be seen in *D3 Sign in* and *D4 Main Menu*. The version number is in the 'major.minor.patch' format.

Major: Intrusive changes

Minor: Non-intrusive changes

Patch: Fixes

It is recommended to consult the manual after a major increment, and you may also wish to do this following a minor increment.

D2 LOADING, ERRORS AND DIALOGS



FIGURE 7. LOADING SPINNER.

The loading spinner in *Figure 7* will appear within the app to indicate that a particular screen is loading. If you have problems with the loading page, see *E Troubleshooting*.

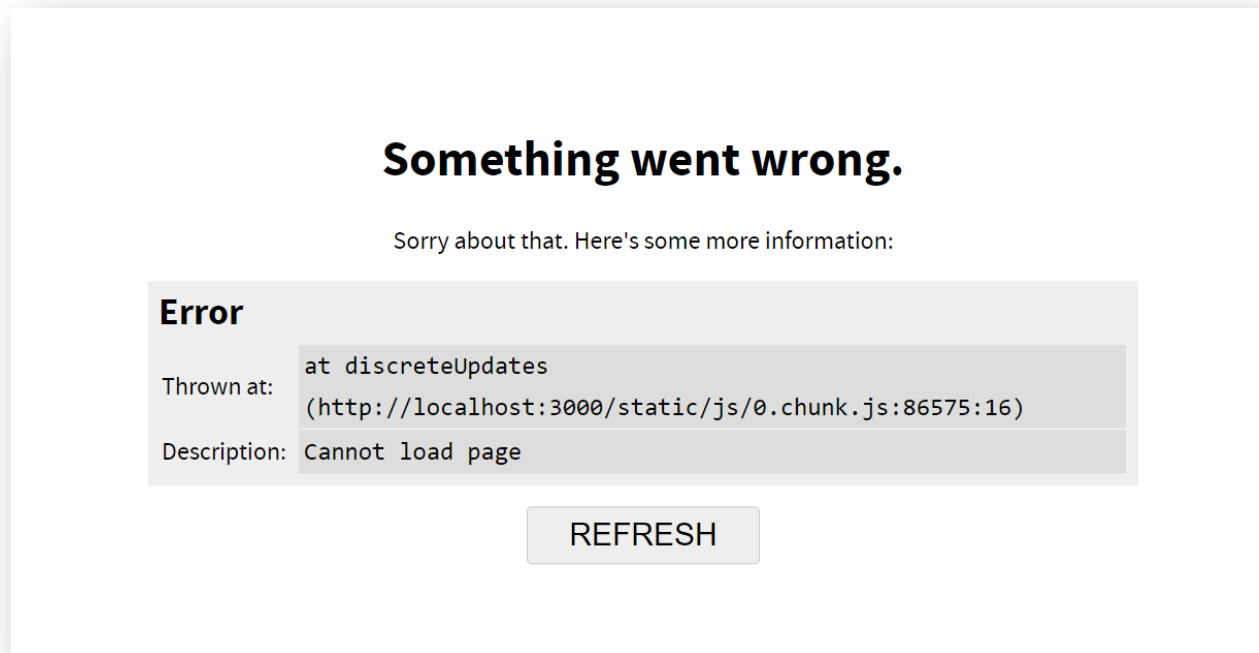


FIGURE 8. ERROR SCREEN WITH DESCRIPTION.

The error screen in *Figure 8* appears when TrayMaster encounters a critical error from which it cannot recover. Try tapping the refresh button to reload TrayMaster. If this does not help, try reinstalling the TrayMaster PWA (see *D1.2 Installing TrayMaster*) or contacting your system administrator. The error message you see may be useful to the administrator.

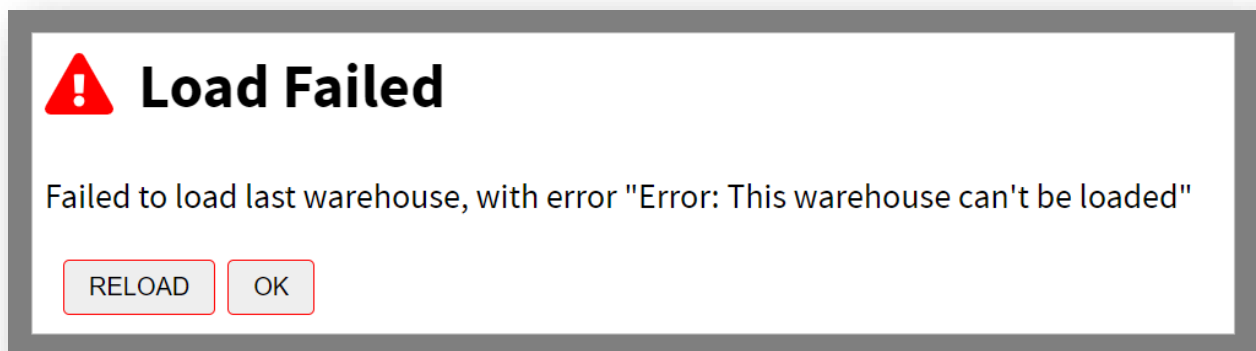


FIGURE 9. ERROR DIALOG TO NOTIFY THAT THE WAREHOUSE FAILED TO LOAD.

The dialog in *FIGURE 9* is sometimes also displayed when a critical error occurs; and may be useful to pass to your system administrator.



FIGURE 10. DIALOG TO CONFIRM A DELETION.

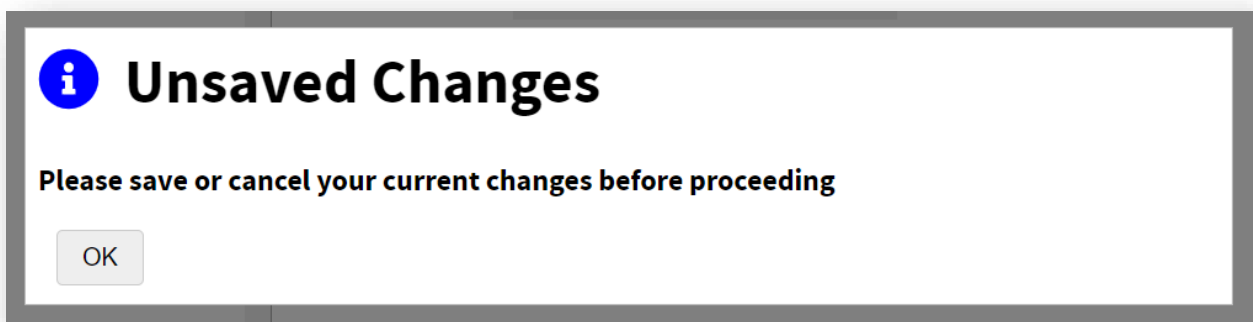


FIGURE 11. DIALOG TO NOTIFY ABOUT UNSAVED CHANGES.

Other dialogs like those seen in *Figure 10* and *Figure 11* can be seen throughout the app.

To confirm a deletion in *Figure 10*, tap 'Delete', and to cancel without deleting tap 'Cancel'.

In *Figure 11*, you can tap 'OK' to close the dialog. You must then save or cancel your changes to prevent this dialog appearing again.

D3 SIGN IN

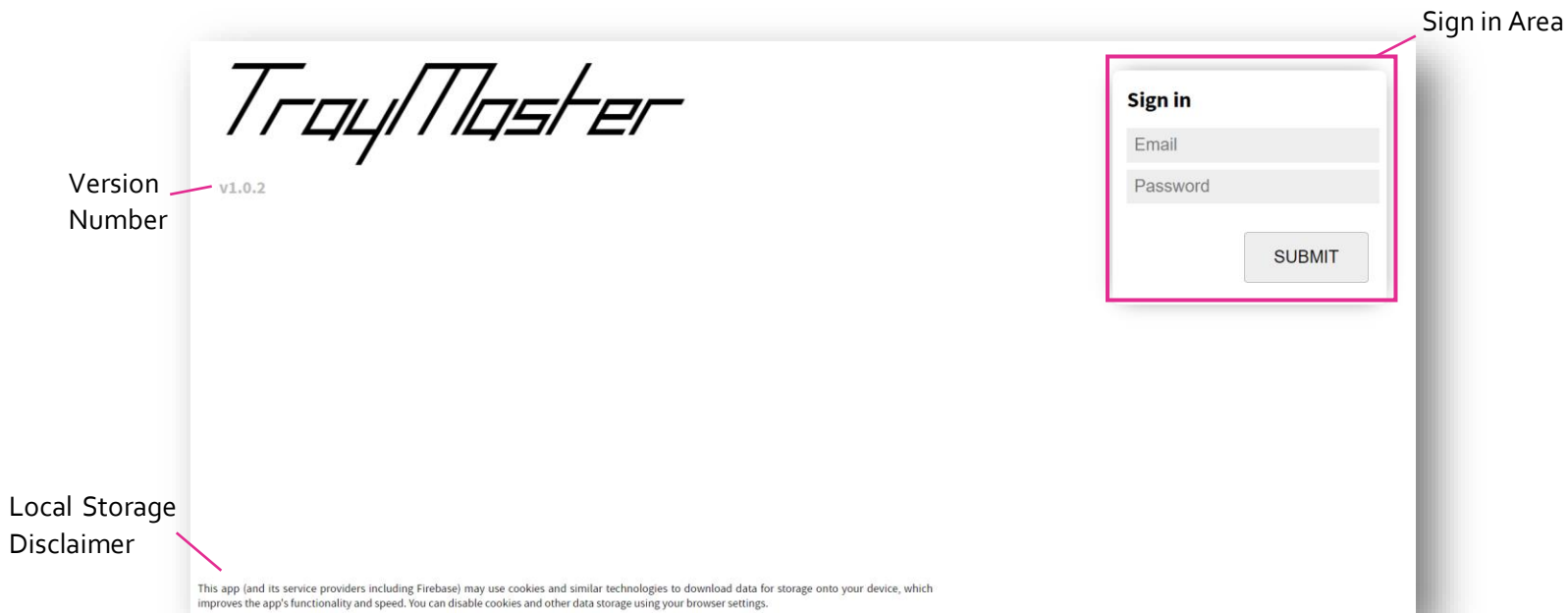


FIGURE 12. SIGN IN SCREEN.

If you have not yet authenticated, this is the first screen visible when accessing TrayMaster. It allows you to **sign in** to your TrayMaster account. It also displays the app's **version number** and the **local storage disclaimer**.

The local storage disclaimer is a privacy notice to inform you about the conditions of using TrayMaster and **may change** in later versions. Please familiarise yourself with the disclaimer before signing in.

D3.1 SIGN IN AREA

The sign in area in the top right allows you to enter your credentials and sign in. If there are any issues when signing in, an error message will be displayed. If you cannot sign in, see *E Troubleshooting*.

D4 MAIN MENU



FIGURE 13. MAIN MENU.

The main menu allows you **navigate** to other screens of TrayMaster, to **sign out** and to **switch warehouse**.

On the **left side**, you can use the navigation buttons to switch to a different screen.

- Shelf View – view and edit the warehouse contents
- Find – perform searches on the warehouse contents to aid with replenishing
- Settings – view and edit your personal settings and the warehouse settings

For full guides see *D6 Shelf View*, *D7 Find*, and *D8 Settings*.

The current user and warehouse are visible in the **bottom right**.

- Tapping the **sign out** button will sign you out, and redirect you to the sign in screen (see *D3 Sign in*)
- Tapping the **switch warehouse** button will redirect you to the switch warehouse screen (see *D5 Switch Warehouse*)

D5 SWITCH WAREHOUSE



FIGURE 14. SWITCH WAREHOUSE SCREEN.

This screen allows you to select a different warehouse from the Warehouse List. You can select any of the warehouses that your user account is associated with.

D6 SHELF VIEW

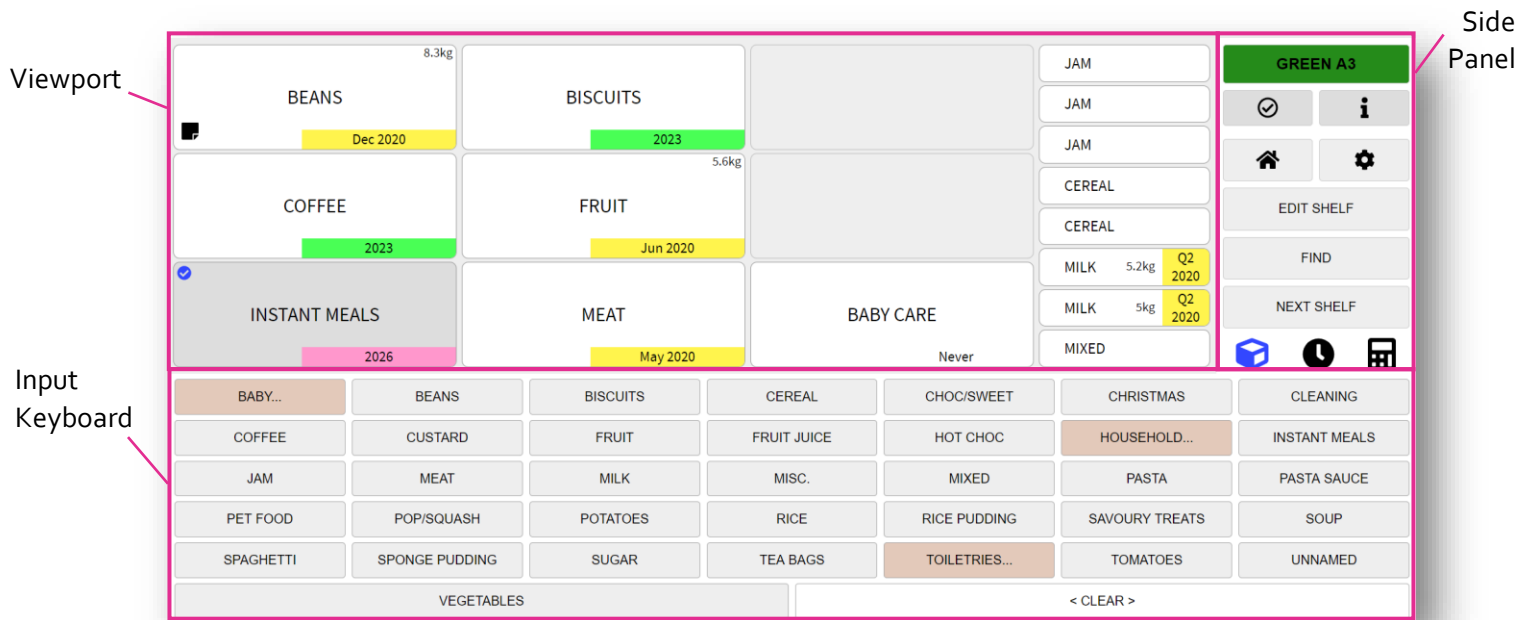


FIGURE 15. SHELF VIEW SCREEN, SHOWING AN EXAMPLE SHELF.

Shelf View is the main screen of TrayMaster. It lets you navigate around the warehouse, edit shelves' dimensions and input data onto their trays.

If a loss of internet connection occurs while on this page, TrayMaster attempts to queue and upload changes when a connection is restored. If a loss of connection persists to a point at which the application cannot continue to function, an error message will be shown.

When you return to shelf view after leaving, your current shelf will be restored.

The screen is split into 3 sections:

- the current **viewport**
- the **input keyboard**
- the **side panel**

The main features of this screen are:

- **navigating** the warehouse
- **selecting** trays
- **inputting data** using the **keyboards**

D6.1 SIDE PANEL

RELATING TO NAVIGATION, SELECTION, EDIT SHELF, FIND AND INPUT KEYBOARDS

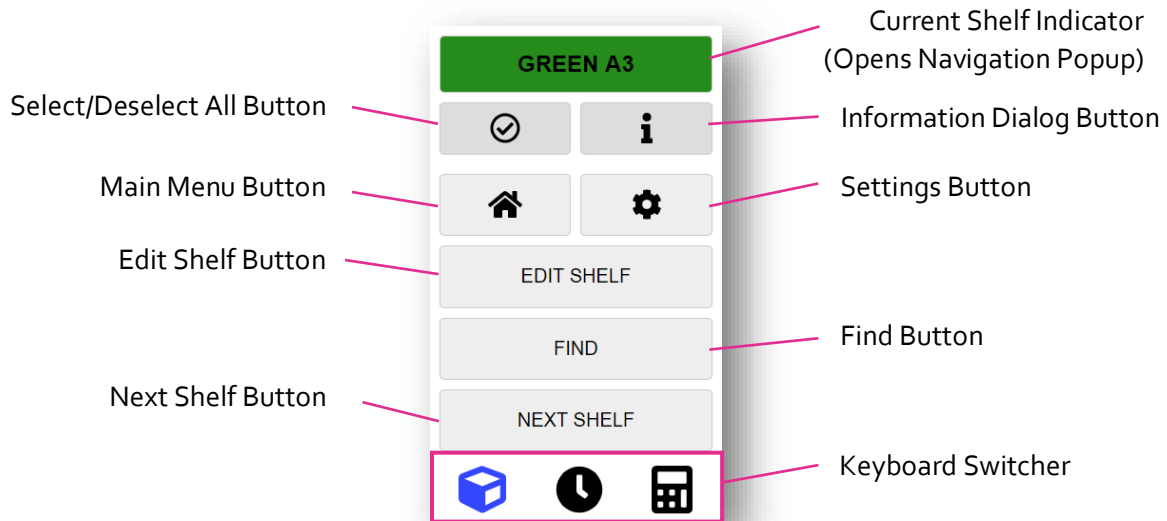


FIGURE 16. THE SIDE PANEL

The side panel area contains several essential shelf view functions, as well as buttons which allow you to navigate TrayMaster.

The side panel contains the following features:

- **Main Menu Button**, this button takes you to the main menu page, see *D4 Main Menu*.
- **Settings Button**, this button takes you to the settings page, see *D8 Settings*.
- **Find Button**, this button takes you to the find page with a preset query depending on your current selection.
 - If your selection is empty, then find will perform a search on all trays
 - If you have trays selected, then find will perform a search for the categories within your selection

Features explained in depth elsewhere:

- **Current Shelf Indicator**, see *D6.8 Navigation*.
- **Select/Deselect All Button**, see *D6.3 Selection*.
- **Information Dialog Button**, see *D6.6 Tray Information Dialog*
- **Edit Shelf Button**, see *D6.7 Edit Shelf*
- **Next Shelf Button**, see *D6.8 Navigation*
- **Keyboard Switcher**, see *D6.4 Input Keyboards*

D6.2 VISUALISATION

RELATING TO VIEWPORT

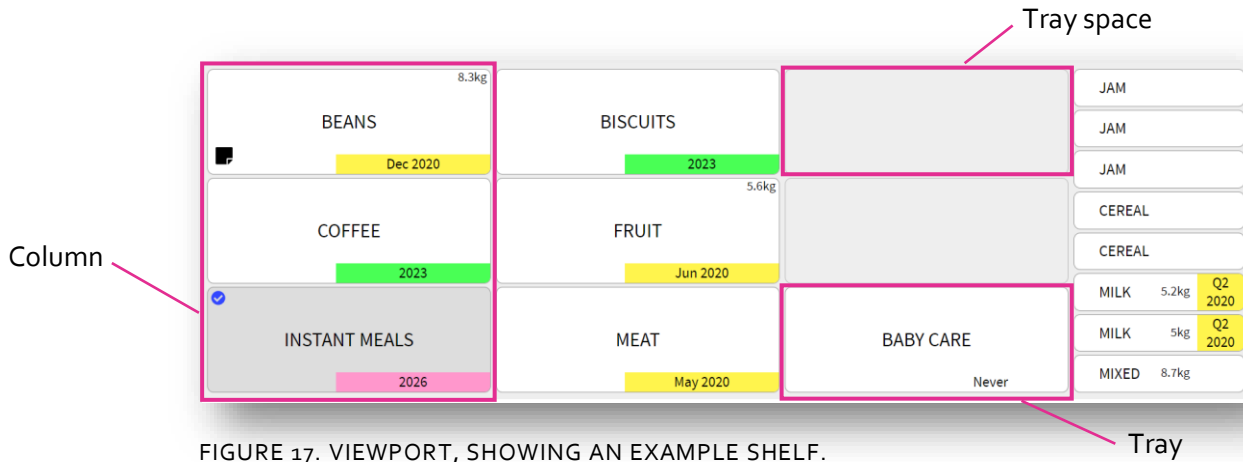


FIGURE 17. VIEWPORT, SHOWING AN EXAMPLE SHELF.

The viewport always shows a single shelf of the warehouse.

Figure 17 shows an example shelf with four columns. The first three columns have a maximum height of 3 trays, and the last column has a maximum height of 8 trays. You can edit the dimensions of a shelf to match your physical shelf's dimensions, see [D6.7 Edit Shelf](#).

In Figure 17, the third column contains one tray, and two empty tray spaces above it. Notably, it is impossible for a tray to exist above a tray space (meaning that trays obey the laws of physics).

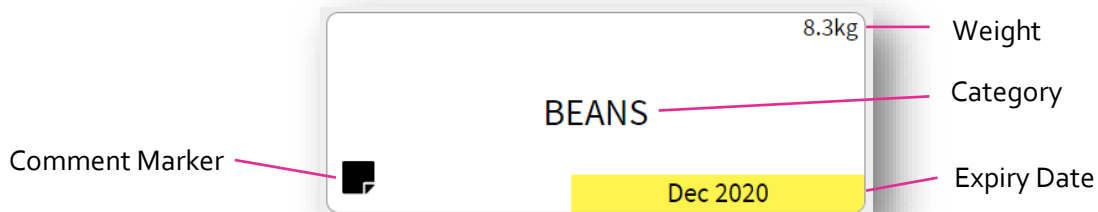


FIGURE 18. AN EXAMPLE TRAY.

Figure 18 shows an example tray, labelled with its important information. If a property is not set, then it will not appear on the tray. The comment marker icon appears on trays which have a comment attached. To view or edit the comment, and to see who last modified the tray and when, see [D6.6 Tray Information Dialog](#).

D6.3 SELECTION

RELATING TO VIEWPORT AND SELECTION

Selection is an important part of how you input data on the shelf view screen. A tray in the viewport must be selected before it can be edited using the input keyboards.

When a tray is selected, a blue tick appears in the upper left-hand corner and its background darkens, as shown in *Figure 19*.

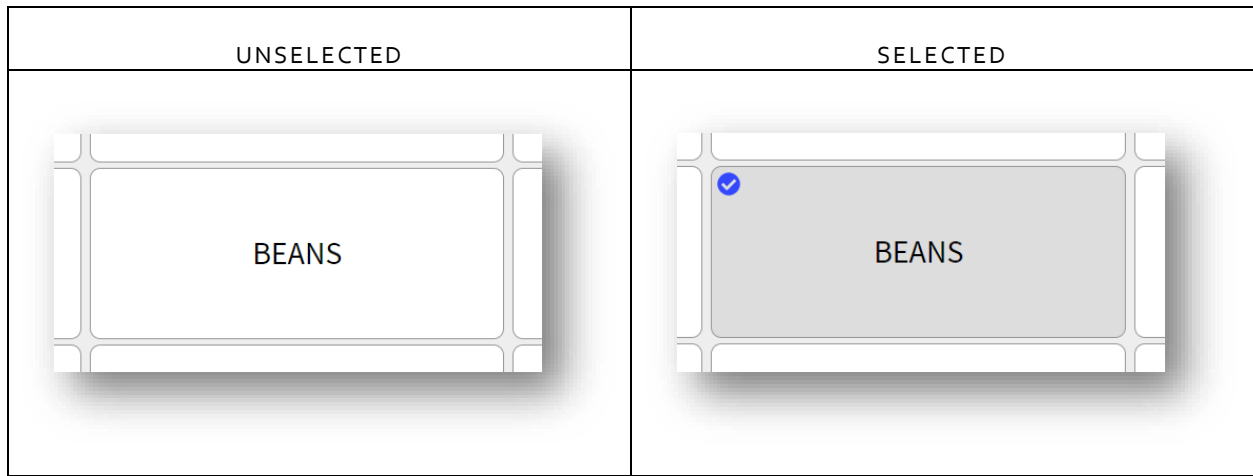


FIGURE 19. DIFFERENCE BETWEEN AN UNSELECTED TRAY AND A SELECTED TRAY.

If there are no trays currently selected, just tap on any tray to select it.

At any point where only one tray is selected, you are in **single-select** mode. In this mode, tapping on another tray will *replace* the current selection. Tapping on the selected tray again will deselect it.

If more than one tray is selected, you are in **multiple-select** mode. In this mode, tapping on an unselected tray will *add* it to the current selection. Greyed-out ticks appear in the upper left of all tray cells to indicate you are in **multiple-select** mode (see *Figure 20*). Tapping on a selected tray again will deselect it. Deselecting trays until only one tray is left will return you to single-select mode.

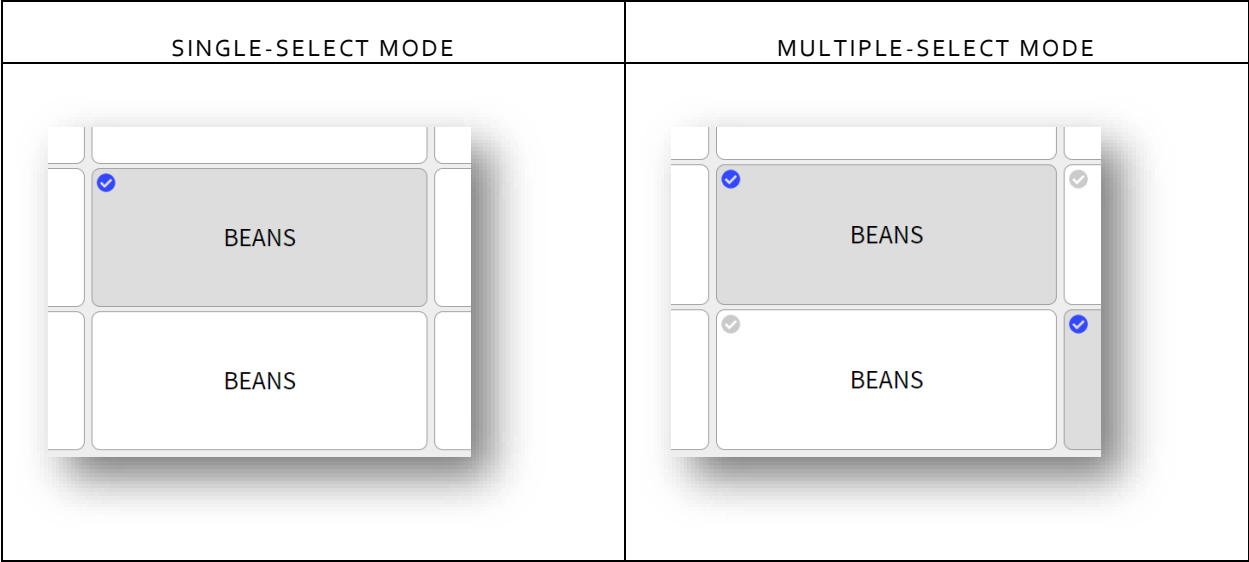


FIGURE 20. DIFFERENCE BETWEEN SELECTION MODES.

You can select more trays (and thus enter multiple-select mode) by tapping and holding on a second tray. Once you are in multiple-select mode, you can select/deselect any additional trays by simply tapping on them.

You can also select more trays using **drag selection**. Tap and hold on a tray until you are visibly in multiple-select mode (greyed-out ticks appear), and then drag to another tray/tray space, without lifting your finger. This selects all trays in between the start and end points of the selection. The trays in between are determined by top-to-bottom, left-to-right ordering (see *Figure 21*).

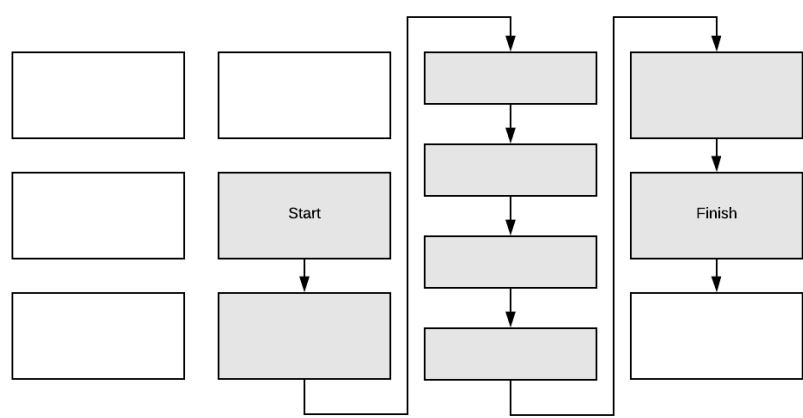


FIGURE 21. DIAGRAM SHOWING THE BEHAVIOUR OF TRAY SELECTION IN DRAG SELECTION.

The icon on the **select/deselect all** button in the side panel will change depending on its behaviour (see *Figure 22*). If some trays are already selected, then it will act as Deselect All. If no trays are selected, it will Select All trays in the viewport.

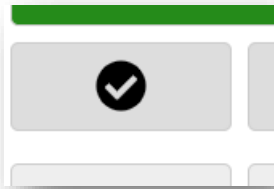
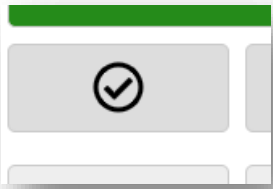
| SELECT ALL | DESELECT ALL |
|---|--|
|  |  |

FIGURE 22. DIFFERENCE BETWEEN SELECT/DESELECT ALL BUTTON ICONS.

D6.4 INPUT KEYBOARDS

RELATING TO INPUTTING DATA

To input data onto the selected trays you must use the corresponding **input keyboards** (category, expiry, and weight). You can use the **keyboard switcher** to switch between them (see *Figure 23*). The current keyboard is selected in blue.

D6.4.1 CATEGORY KEYBOARD

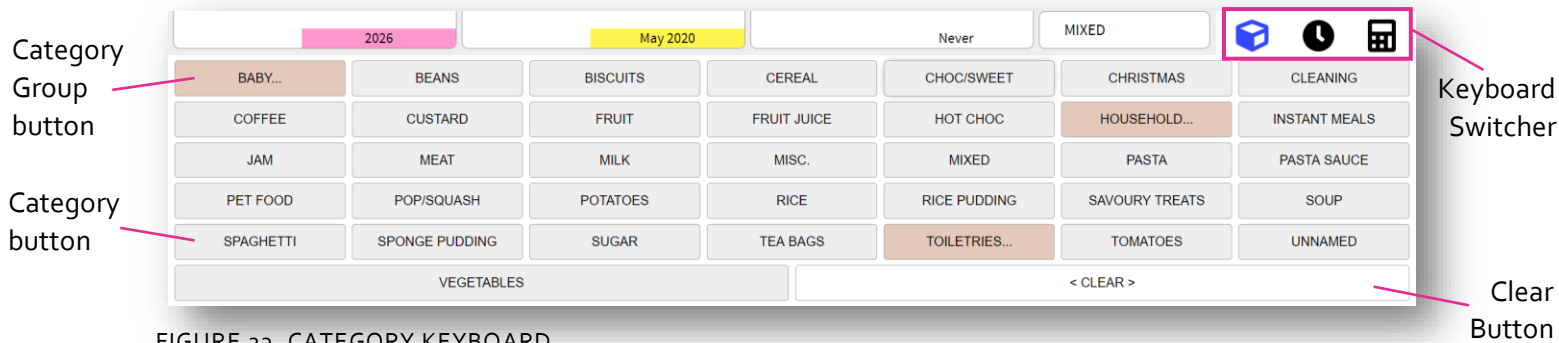


FIGURE 23. CATEGORY KEYBOARD.

The category keyboard in *Figure 23* is used to assign a category to selected tray(s). If no trays are selected, the buttons on this keyboard are disabled. Tap a category button to apply it to the selected tray(s). When a category is assigned to a tray, all of its other properties are cleared. Use the Clear button to replace all selected trays with empty tray spaces.

Buttons with a gold background are category group buttons. Tapping these will open a popup that displays all of the category buttons within that group, see *Figure 24*. Category buttons in this popup function the same as the regular category buttons directly on the input keyboard. You can close the popup by choosing one of the categories or tapping the screen outside of the popup.



FIGURE 24. CATEGORY GROUP POPUP, SHOWING THE CATEGORY BUTTONS WITHIN THAT GROUP.

To edit category titles and choose how categories are grouped, see *D8.2 Category Editor*.

D6.4.2 EXPIRY KEYBOARD

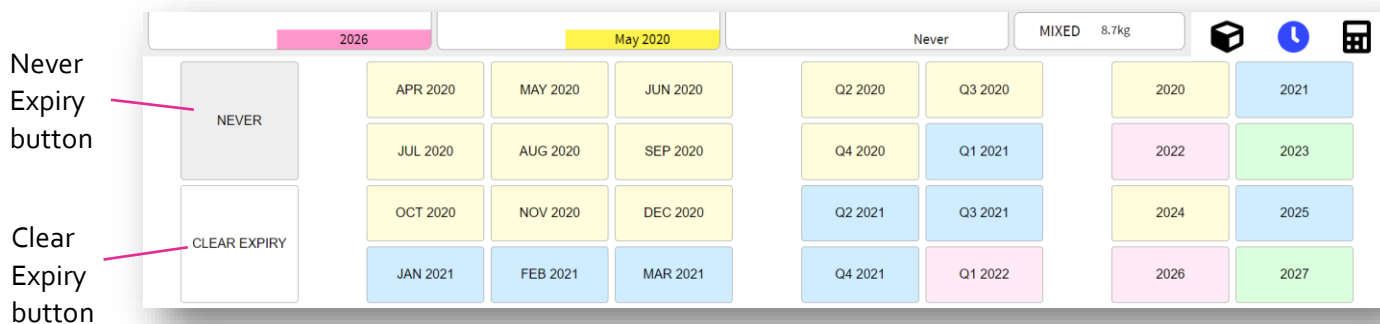


FIGURE 25. EXPIRY KEYBOARD.

The expiry keyboard in *Figure 25* is used to assign an expiry to selected tray(s). If no trays are selected, the buttons on this keyboard are disabled. When the expiry buttons are pressed, the corresponding expiry range is added to the tray (unlike applying a category which clears all other properties). The "Never" expiry button is used to denote that a tray never expires. The Clear Expiry button only clears any expiry from the tray – unlike the Clear button on the Category keyboard (which clears all properties).

The expiry buttons (except for the Never expiry) are colour coded in a four-year loop which corresponds to the expiry label on the trays. In the coming year, the expiry can be specified down to the month. In the two coming years, the expiry can be specified down to the quarter.

D6.4.3 WEIGHT KEYBOARD

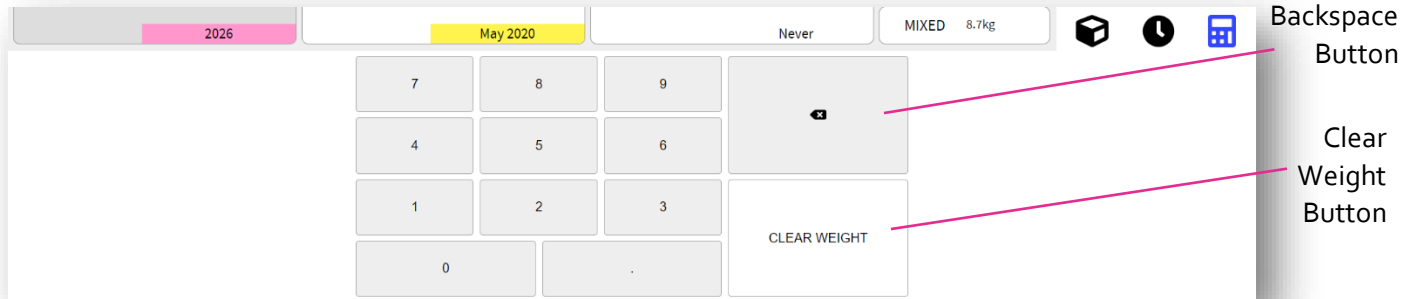


FIGURE 26. WEIGHT KEYBOARD WITH AUTO-ADVANCE OFF.

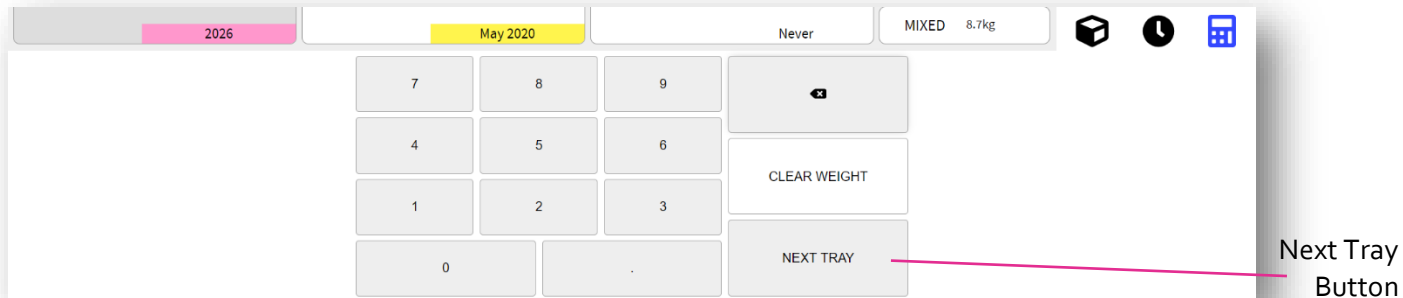


FIGURE 27. WEIGHT KEYBOARD WITH AN ACTIVE AUTO-ADVANCE MODE.

The weight keyboard shown in *Figure 26* and *Figure 27* is used to assign a weight to selected tray(s).

Use the number and decimal point buttons to type a weight to apply to the selected tray(s). If you make a mistake, you can tap the backspace button to delete the last character in the weight field. The Clear Weight button only clears the selected trays' weights – unlike the Clear button on the Category keyboard (which clears all properties).

If an auto-advance mode has been selected, the **Next Tray** button will also be displayed on this keyboard; see *Figure 27*. Tapping this button will advance your selection from the currently selected tray to the next one, moving bottom-to-top in the column and from left-to-right through the columns. For a full guide on this, see *D6.5 Auto-advance*.

D6.4.4 UNIFIED KEYBOARD (BETA)

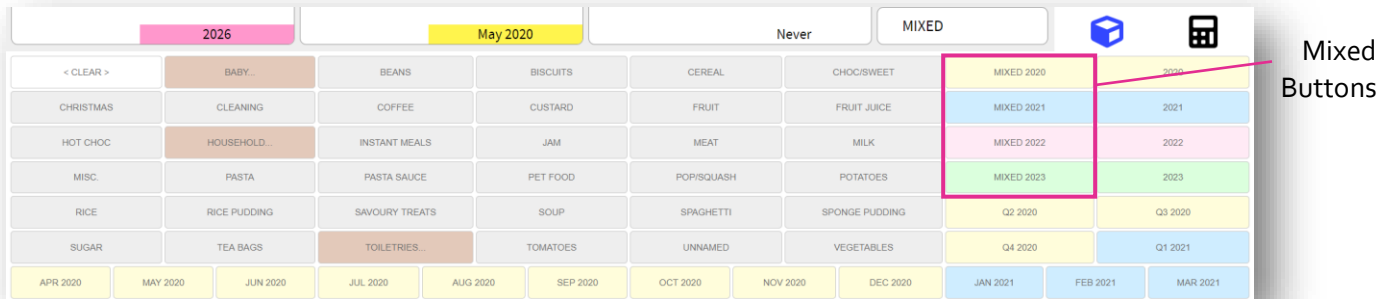


FIGURE 28. UNIFIED KEYBOARD.

You can enable this experimental keyboard in your settings, see *D8.1 Personal Settings*.

WARNING: *This is a beta feature and may not function as expected. It can only display up to 35 category buttons, and any additional categories in the warehouse will not be shown.*

The category and expiry buttons on this keyboard behave identically to those on the separate category and expiry input keyboards. See *D6.4.1 Category Keyboard* and *D6.4.2 Expiry Keyboard*.

The clear button behaves identically to the clear button on the category keyboard (clearing *all properties* on the selected trays).

The **Mixed buttons** are unique to this keyboard. With one button, you can set selected trays' category to "Mixed" and set their expiry to the desired year. These buttons will also clear any existing comment or weight set on the trays.

D6.5 AUTO-ADVANCE

RELATING TO SELECTION AND KEYBOARDS

Auto-advance is a feature for automatic selection advancing and automatic keyboard advancing. It provides faster data entry by removing the need to manually change which trays are selected and to switch between input keyboards. Before reading this section, see *D6.3 Selection* and *D6.4 Input Keyboards*.

Auto-advance can operate in one of several modes. The input keyboards will automatically advance to collect the required information about the currently selected tray(s), according to the sequence selected in the settings (see *D8.1 Personal Settings*):

- Category → Expiry → Weight
- Category → Expiry
- Weight

If you are using the experimental unified keyboard, it will not automatically advance until all the required data has been collected. For example, in the first two modes it will wait for both category and expiry data.

Once all required properties are collected, the selection advances to the next tray (and the input keyboard starts again in its sequence). The next tray is defined in an ordering of bottom-to-top and left-to-right, see *Figure 29*.

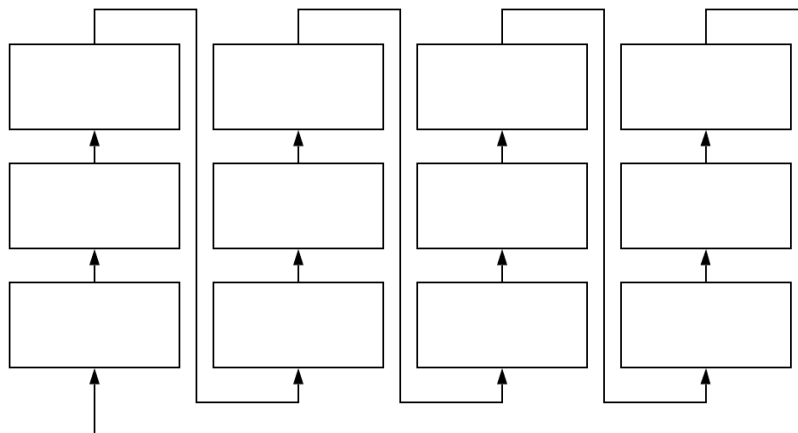


FIGURE 29. DIAGRAM SHOWING HOW THE **NEXT TRAY** IS DECIDED FOR AUTO-ADVANCE.

If multiple trays are selected, the next tray is defined as the next tray of the last tray in this ordering. You can disable auto-advance with multiple trays selected, see *D8.1 Personal Settings*.

D6.6 TRAY INFORMATION DIALOG

RELATING TO SIDE PANEL AND SELECTION

You can tap on the Tray Information button in the side panel (see *Figure 30*) to open this dialog.



FIGURE 30. TRAY INFORMATION BUTTON IN THE SIDE PANEL.

If **exactly one tray** is selected, this dialog will display the tray's current comment, the last time the tray was edited and by whom; see *Figure 31*.



FIGURE 31. TRAY INFORMATION DIALOG WITH ONE TRAY SELECTED.

You can use the text field to edit the comment, and then Save or Cancel by using the relevant buttons.

If **multiple trays** are selected and the Save button is pressed, the comment will be applied to all selected trays – and if any of the selected trays had a pre-existing comment, it will be overwritten. When multiple trays are selected, you cannot view the Last Modified information (see *Figure 32*).

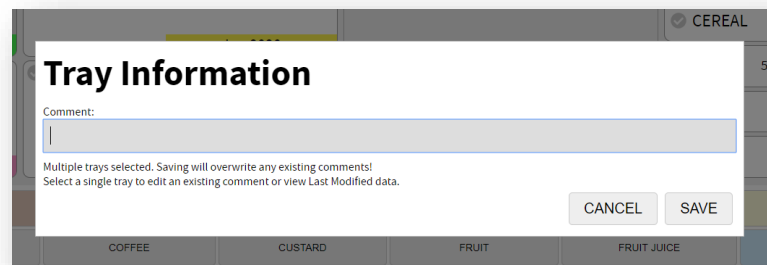


FIGURE 32. TRAY INFORMATION DIALOG WITH MULTIPLE TRAYS SELECTED.

D6.7 EDIT SHELF

RELATING TO SIDE PANEL AND VIEWPORT

Tapping on the Edit Shelf button in the side panel will enter the Edit Shelf screen.

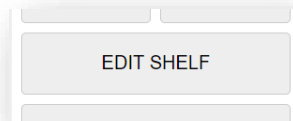


FIGURE 33. EDIT SHELF BUTTON IN THE SIDE PANEL.

In this screen, the viewport is overlaid with controls corresponding to each of the shelf's columns. You can use them to delete a column, adjust its height, and adjust its width. By using the controls, you can ensure that the current shelf and its columns match the physical world, and thus model your warehouse more accurately.

WARNING: *Changes made on this screen are applied immediately; you cannot undo them.*

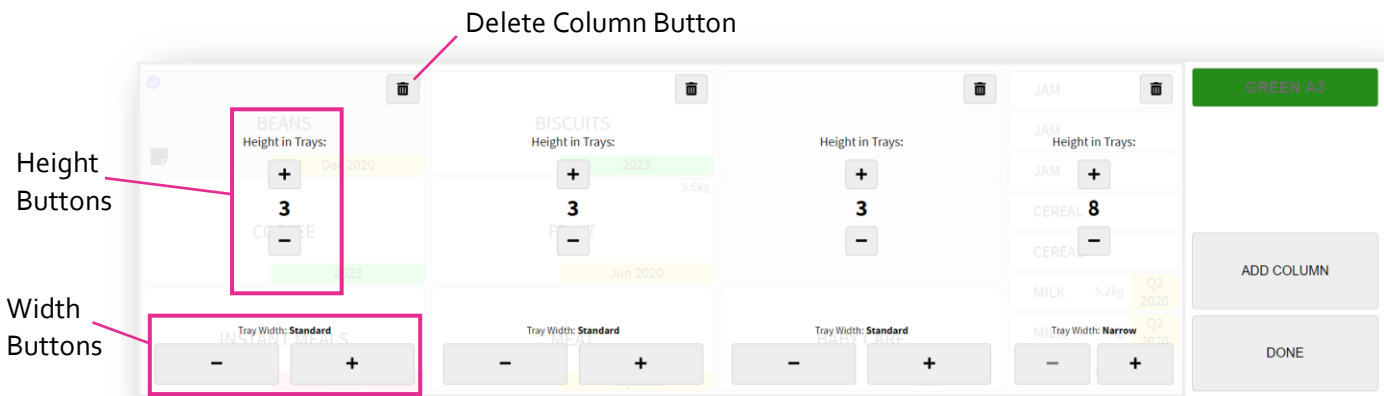


FIGURE 34. VIEWPORT IN EDIT SHELF SCREEN.

Functionality offered by edit shelf mode, as seen in *Figure 34*:

- Use the **height buttons** to increase/decrease the height (trays and tray spaces in total) of a column.
- Use the **width buttons** to adjust the visual width of the column. Choose from Narrow, Standard and Wide, see *C1 Warehouse Modelling*.
- Use the **delete column button** to remove a column.
- Click the '**Add Column**' button to insert a new column on the right side of the shelf.

Once you have finished editing, click '**Done**' to leave Edit Shelf.

D6.8 NAVIGATION

RELATING TO SIDE PANEL AND NAVIGATION

Tapping on the current shelf indicator in the side panel will open the Navigation Popup.



FIGURE 35. CURRENT SHELF INDICATOR IN THE SIDE PANEL.

This popup visualises the current zone, allows you to change zone, and helps you to navigate around your warehouse.

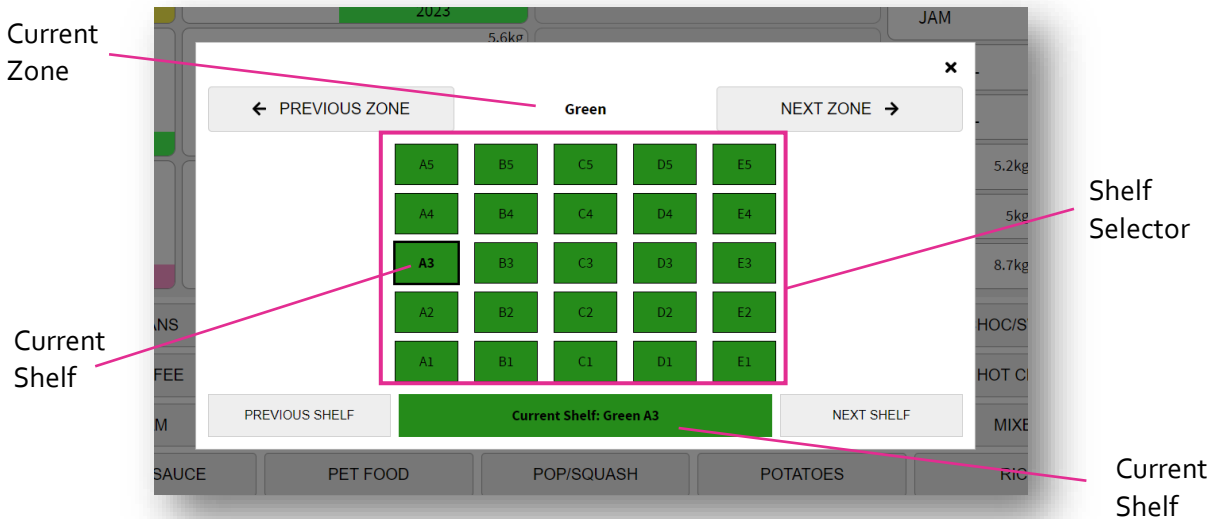


FIGURE 36. THE NAVIGATION POPUP.

- Use the 'Previous Zone' and 'Next Zone' buttons at the top of the popup to choose a zone. The name of the current zone is displayed in the centre. The ordering of the zones is the same as in the Zone Editor screen (see *D8.3 Zone Editor*).
- The **shelf selector** visualises the current zone. Tap on any shelf to navigate directly to it. The current shelf is marked with a thick black border.

You can also use the buttons at the bottom of the popup to navigate between shelves. The name of the current shelf is displayed in the centre. The ordering of shelves is shown in *Figure 37*. Essentially, the next shelf ordering is bottom-to-top within each bay, left-to-right through the bays in the current zone, and then onto the next zone. The previous shelf is simply the inverse operation of the next shelf.

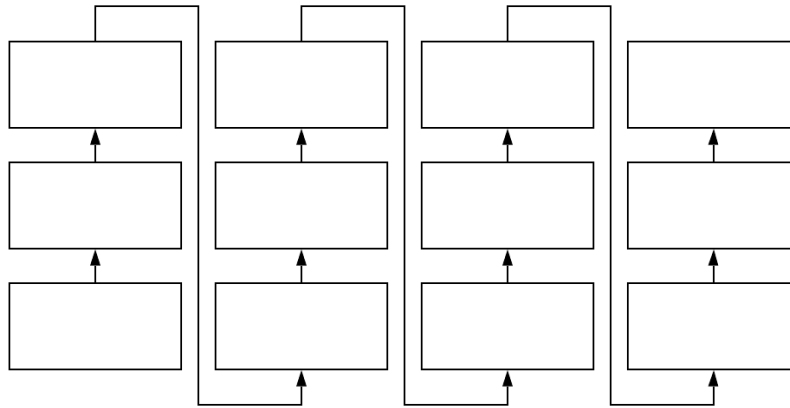


FIGURE 37. DIAGRAM SHOWING HOW THE **NEXT SHELF** IS DECIDED WITHIN A ZONE.

The side panel contains an equivalent 'Next Shelf' button, and you can configure it to display a 'Previous Shelf' button too (see *D8.1 Personal Settings*). *Figure 38* shows how these buttons are displayed in the side panel.

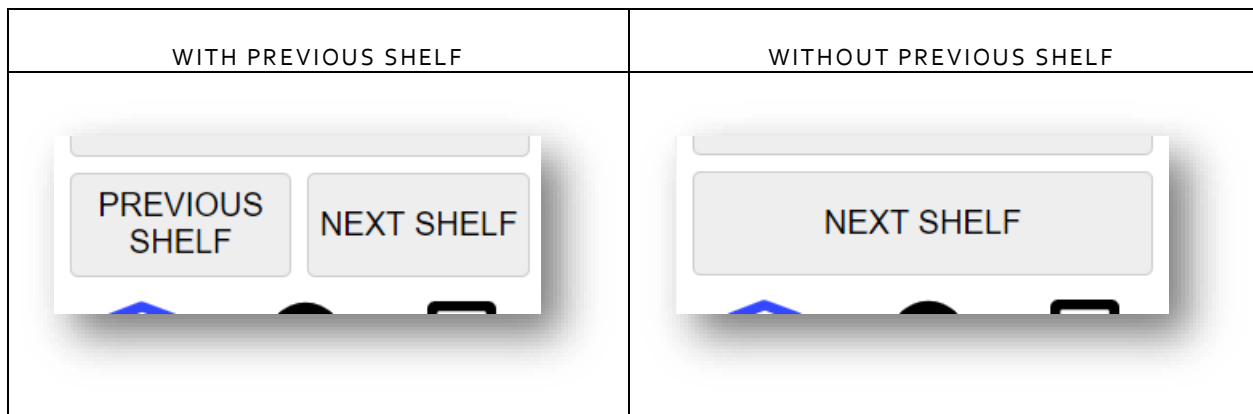


FIGURE 38. THE SIDE PANEL WITH AND WITHOUT THE PREVIOUS SHELF BUTTON.

D7 FIND



FIGURE 39. A FIND QUERY WITH MULTIPLE CATEGORIES SELECTED.

Find is used to locate the trays in your current warehouse that belong to particular categories. Use the category buttons on the right side of the screen to form your query. The results are displayed on the left side of the screen.

When using find, as seen in *Figure 39*:

- The **'Back Button'** returns you to your previous screen.
- The **'Find Sentence'** summarises your current query. Click the cross at the end of the sentence to clear your current query and search for all categories instead.
- The **'Download'** button allows you to export the results table as a comma-separated values (CSV) file onto your device.
- The **'Category buttons'** allow you to select and deselect the categories that form your current query. You can select up to 10 categories at once.
- The **'Results table'** lists the trays that belong to any of your selected categories. It displays all relevant data including expiry, weight, location and comment. The expiry and zone information are colour-coded in the same way as displayed in Shelf View. Tapping the location will take you to the corresponding shelf.

D8 SETTINGS

The settings screen allows you to personalise the way that you use TrayMaster, as well as edit the categories and zones within your warehouse.

You can switch between the settings page tabs by tapping on the corresponding labels on the left side of the screen.

All settings under the Personal Settings tab will only apply to your user account, and do not affect other users. Conversely, the Category Editor and the Zone Editor are only available to administrator users, because categories and zones are shared by all users in a warehouse. Any edits made under those tabs will affect other users in the warehouse.

D8.1 PERSONAL SETTINGS

| Settings | Personal Settings |
|-------------------|---|
| User | Shelf View |
| Personal Settings | Show Previous Shelf button <input type="checkbox"/> |
| Warehouse | Clear all trays above when clearing <input checked="" type="checkbox"/> |
| Category Editor | Auto-advance mode Category > Expiry > Next Tray |
| Zone Editor | Auto-advance with multiple trays selected <input checked="" type="checkbox"/> |
| | Use unified keyboard (beta) <input checked="" type="checkbox"/> |

FIGURE 40. PERSONAL SETTINGS SCREEN.

In personal settings you can configure the behaviour of the TrayMaster app for you.

Remember: these settings are **personal** to your account.

When using personal settings, as seen in *Figure 40*:

- **'Show Previous Shelf button'** enables the optional "Previous Shelf" button in the side panel of the Shelf View screen. See *D6.8 Navigation*.
- **'Clear all trays above when clearing'** means that when a tray is cleared, all trays above it in the column will also be cleared.

- In the '**Auto-advance mode**' dropdown menu, you can either select an auto-advance mode or turn this feature Off. If you do select an auto-advance mode, then the app will automatically advance the input keyboards in the specified sequence without requiring manual switching of the keyboards. Once all required data has been collected on one tray, the app will automatically advance to the next tray. See *D6.5 Auto-advance* for more information.
- If '**Auto-advance with multiple trays selected**' is disabled, then auto-advance has no effect in multiple-select mode.
- '**Use unified keyboard (beta)**' merges the separate category and expiry input keyboards into a singular experimental unified keyboard. The weight keyboard remains a separate input keyboard. See *D6.4.4 Unified Keyboard (beta)* for more information.

D8.2 CATEGORY EDITOR

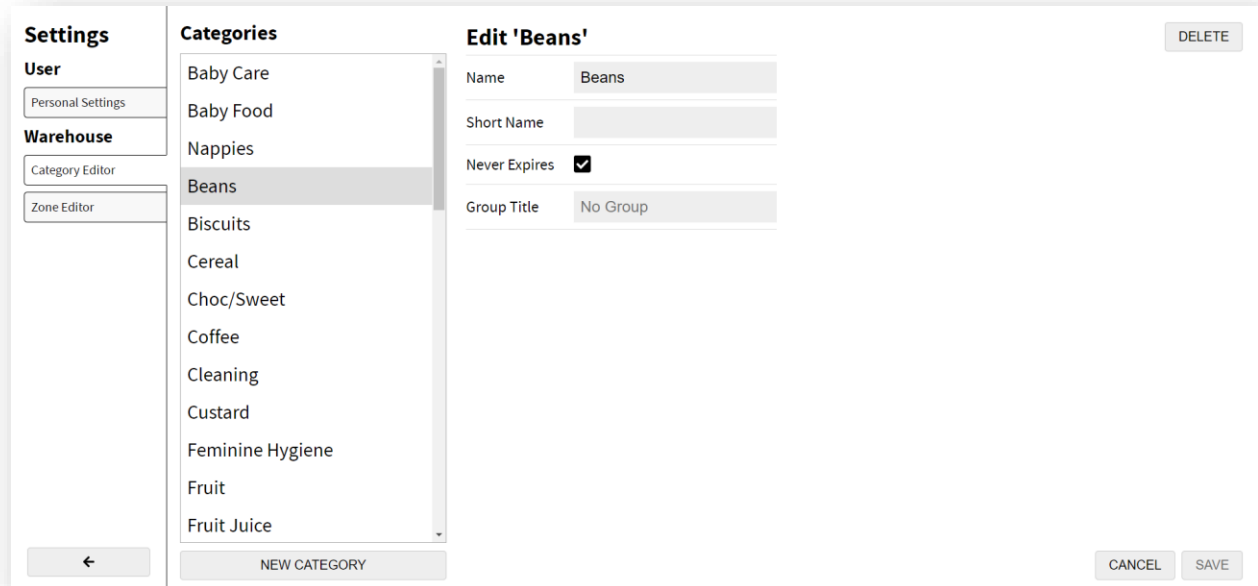


FIGURE 41. THE CATEGORY EDITOR SCREEN.

The category editor allows you to edit, create and delete categories in the warehouse. You can tap an existing category in the list to edit it, or you can create a new category by tapping the New Category button.

Remember: categories are **shared** by all users in a warehouse, so your changes will affect others.

When using the category editor, as seen in *Figure 41*:

- **'Name'** is the category's main identifier. It is displayed on trays visualised in shelf view, on the category input keyboard, and in other screens throughout the app.
- If the **'Short Name'** is set, this will be displayed on the category keyboard instead of the full category name.
- By enabling **'Never Expires'**, you can ensure that any trays which are assigned this category using the category keyboard automatically have their expiry set to "Never" too. If auto-advance is enabled, then the app will skip the expiry keyboard as it will not be required for that tray.
- With **'Group Title'** you can group multiple related categories together, so they display under one Category Group button on the input keyboard. Enter the same Group Title for all the categories you would like to group together. See *D6.4.1 Category Keyboard*.
- Tapping **'Delete'** will open a dialog for you to confirm or cancel the deletion. Any tray in the warehouse that is currently associated with that category will have its category cleared.

D8.3 ZONE EDITOR

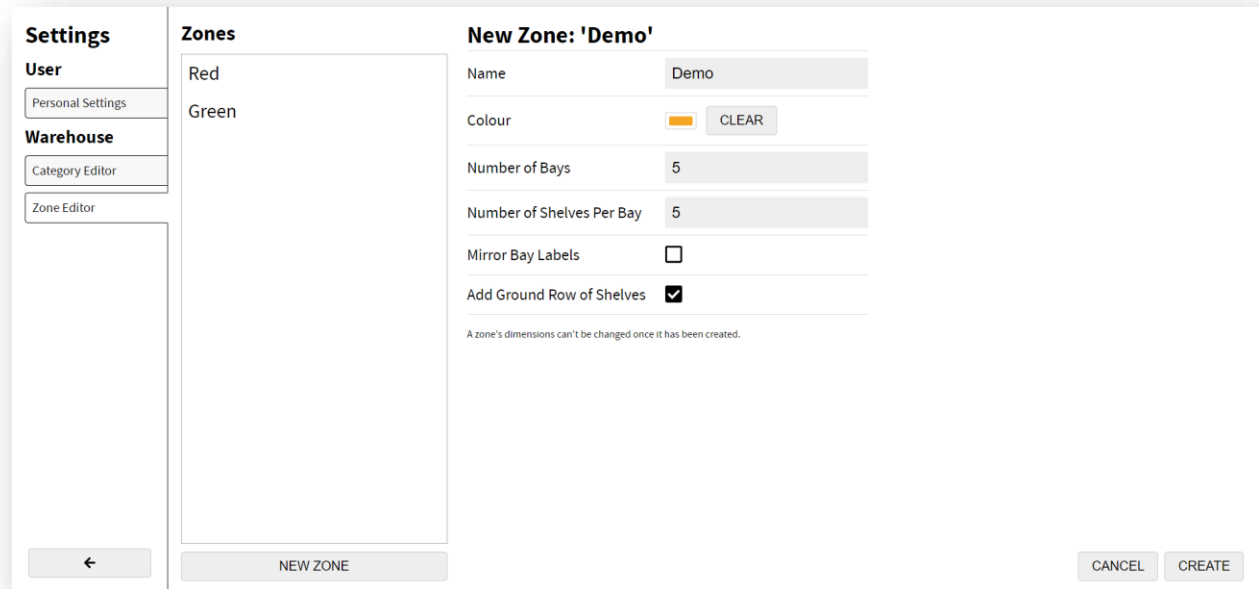


FIGURE 42. THE ZONE EDITOR SCREEN WHEN CREATING A NEW ZONE.

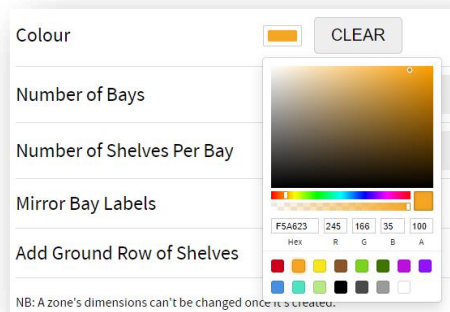


FIGURE 43. SETTING THE COLOUR OF A NEW ZONE.

The zone editor allows you to edit, create and delete zones in the warehouse. You can tap an existing zone in the list to edit it, or you can create a new zone by tapping the New Zone button.

Remember: zones are **shared** by all users in a warehouse, so your changes will affect others.

When creating a zone with the zone editor, as seen in *Figure 42*:

- **'Name'** is the zone's identifier. If no name is entered, the zone will automatically be named 'Unnamed'.
- **'Colour'** is the colour of the zone, which is used as a visual cue throughout the warehouse. You can tap the coloured rectangle to the left of 'CLEAR' to open a colour picker for choosing a colour, see *Figure 43*. 'CLEAR' will reset the colour to white.

- The '**Number of Bays**' and '**Number of Shelves Per Bay**' determine the dimensions of the zone.
- '**Mirror Bay Labels**' changes the order of the bays. By default, the labels start with 'A' on the leftmost bay. Ticking this box will mirror to make 'A' the rightmost bay instead.
- '**Add Ground Row of Shelves**' changes the numbering of the shelves. By default, the bottom row of shelves starts is numbered as row 1. Ticking this box makes the lowest shelf in each bay have a 'G' instead of a 1, signifying 'ground'.

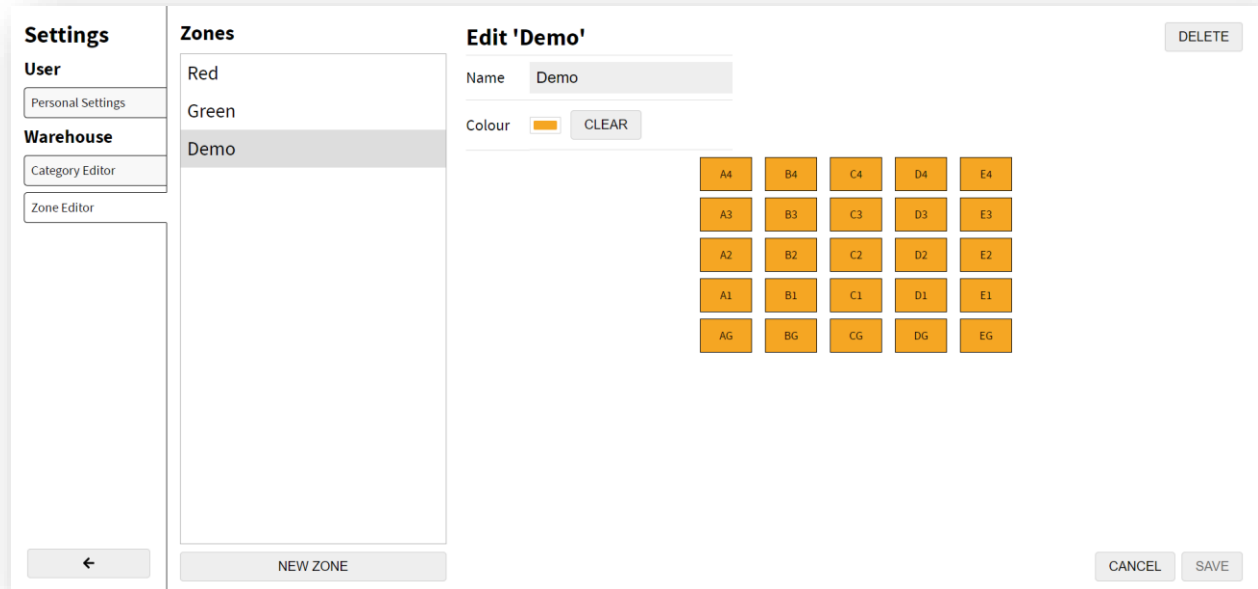


FIGURE 44. THE ZONE EDITOR SCREEN, VIEWING AN EXISTING ZONE.

After a zone has been created, you can only edit its Name and Colour, as seen in *Figure 44*. You cannot edit its dimensions or change the labels of its shelves.

Tapping '**Delete**' will open a dialog for you to confirm or cancel the deletion of the zone.

E TROUBLESHOOTING

| Query | Solution |
|---|---|
| ACCESSING TRAYMASTER | |
| If you see a loading spinner for a prolonged period of time. | Try refreshing the page. |
| If you are permanently stuck on the loading spinner. | Try reinstalling the TrayMaster PWA, see <i>D1.2 Installing TrayMaster</i> . |
| TrayMaster is slow to load after not being used for a while. | This is expected behaviour. See <i>D1 Accessing TrayMaster</i> for more information. |
| The TrayMaster app isn't updating. | You may need to redownload the latest version of the app. Try deleting the app from your device, clearing your browser's cache, reloading the TrayMaster URL and then reinstalling the PWA. |
| SIGN IN SCREEN | |
| You cannot sign in using your username and password. | Contact your system administrator to verify your account details and reset your password if necessary. |
| CHANGE WAREHOUSE SCREEN | |
| You cannot see your desired warehouse. | You may not be authorised to access that warehouse. Contact your system administrator to request that it is added to your list of accessible warehouses. |
| SHELF VIEW SCREEN | |
| The keyboard switcher doesn't show the keyboards that you expect. | You may have enabled the experimental unified keyboard. Go to Settings > User > Personal Settings and switch off 'Use unified keyboard'. |

| | |
|--|---|
| The current shelf has no columns or trays. | The current shelf may be empty. Click Edit Shelf and add some columns (see <i>D6.7 Edit Shelf</i>). |
| After selecting a tray space, it is not possible to input data from the keyboards. | Data cannot be inputted on a tray that is above an empty tray space. |
| SETTINGS SCREEN | |
| You cannot see the Category Editor or Zone Editor. | You may not be authorised to view them. Talk to your system administrator to request that your user privileges are increased. |

F MAINTENANCE AND ADMINISTRATION

THIS SECTION ASSUMES ACCESS TO A CONFIGURED **FIREBASE** PROJECT (IF NOT, CONSIDER G6)

This section will give an outline of how a system administrator can use Firebase to manage accounts, warehouses and user permissions. Firebase is a collection of online cloud services offered by Google.

F1 FIREBASE FEES

Firebase offers free (“Spark”) and ‘pay as you go’ (“Blaze”) plans. For the most up-to-date fee information, visit firebase.google.com/pricing. The following will discuss the limitations with the free plan with non-technical use-case examples. If a fee-based Firebase service is queried after reaching its free access limit (for a particular time period) and you have not provided billing details, then Firebase will refuse the request until the free resource usage is reset for the next billing period. The Firebase authentication services used by TrayMaster have unlimited free usage, however the Cloud Firestore and Firebase Functions have daily or monthly usage limits when using the free plan.

Cloud Firestore involves a collection-document model (where a collection is a set of documents, and a document consists of a set of fields and is the minimum amount of data that can be loaded). Firestore has free daily limits before a fee will be charged (if payment details are configured). These restrict the amount of data stored and transferred, and restrict the number of documents read, written to and deleted as follows:

- Data Storage: 1GiB total (then around \$0.18 per GiB above this)
- Network Egress: 10GiB/month (price varies depending on region above this)
- Document Reads: 50,000/day (then around \$0.06 per 100,000 above this)
- Document Writes: 20,000/day (then around \$0.18 per 100,000 above this)
- Document Deletions: 20,000/day (then around \$0.02 per 100,000 above this)

For the application, data storage and transfer are unlikely to be an issue as warehouses should only be around a few megabytes each. Document *Reads*, *Writes* and *Deletions* can be thought of as follows:

When signing in, the app makes 2 *Read* requests to fetch the user and warehouse settings. Then if the selected warehouse has n_{cat} categories, n_z zones, n_b bays and n_s shelves; the app makes $n_{cat} + n_z + n_b + n_s$ *Read* requests. Trays are then loaded dynamically (when their shelf is selected or moved to). When a shelf with n_c columns and n_t trays is loaded, the app makes $n_c + n_t$ *Read* requests. When using the ‘Find’ feature (or CSV-export cloud function), if n_{find} trays match the query then n_{find} *Read* requests are made. This means to load a whole warehouse, about $2 + n_{cat} + n_z + n_b + n_s + n_s(n_c + n_t)$ *Read* requests are made (where $n_c + n_t$ use averages). For example, downloading a whole warehouse (with 20 categories) of 10 zones with an average of 5 bays per zone, 5 shelves per bay, 4 columns per shelf and 3 trays per column will produce around $2 + 20 + 10 + 50 + 250 + 1000 + 3000 \approx 4000$ *Reads*.

Every time a user setting, warehouse setting, zone or tray is changed, one *Write* request is made.

If an element of the warehouse model is deleted, the app must delete all elements below it too. Each of these also requires a *Delete* request. As such, deleting a column requires a *Delete* request for the column, and for all the trays it contains. Deleting a warehouse requires a *Delete* request for the warehouse plus one for every zone, bay, shelf, column and tray within.

The limitations applied to the HTTP CSV-export cloud function are not an issue as this is the only cloud function and will likely only be used rarely (once or twice per month).

F2 FIREBASE CONSOLE

The Firebase Console is the main point for back-end administration. System administrators will find it useful for viewing/editing users and warehouses.

An existing system administrator must grant access for other Google accounts to use the console. Once access is gained to a project console, it can be accessed at console.firebase.google.com. Once the project is entered, the user is presented with an overview of the project including key statistics such as service usage and function errors.

F2.1 WAREHOUSE AND USER MANAGEMENT

This section will consider how to administrate users and warehouses. Where appropriate, instructions intended for initial user and warehouse additions (on an empty Firebase project) are also provided.

F2.1.1 ADDING NEW WAREHOUSES

1. On the Firebase console, open the *Database* tab.
2. Under the *Data* tab, click *Start collection*.
3. In the popup (under *Collection ID*), type “warehouses” and click *Next*.
4. Under *Document ID*, either press *Auto-ID* (for a randomly assigned warehouse ID) or type a unique ID (used to reference the warehouse).
5. Finally, add a single entry under *Field* of “name” (with a default *Type* of “string”), set *Value* to the name of the Warehouse.
6. To add more warehouses, click on the “warehouses” collection, click *Add document* and perform steps 4 and 5.

F2.1.2 ADDING NEW USERS

1. On the Firebase console, open the *Authentication* tab.
2. Under the *Users* tab, click *Add User*.
3. Enter the respective email and password for the user, then click *Add user*.
4. The user list shows the unique *User UID* assigned to the user, click the *Copy UID* icon.
5. Now open the *Database* console tab.
6. Under the *Data* tab, click *Start collection*.
7. In the popup (under *Collection ID*), type “users” and click *Next*.
8. Paste the previously copied UID under *Document ID*.

9. Finally, add a single entry under *Field* of “name” (with a default *Type* of “string”), set *Value* to the name of the user.
10. To add more users, click on the “users” collection, click *Add document* and perform steps 8 and 9.

F2.1.3 RESETTING USER PASSWORDS

1. On the Firebase console, open the *Authentication* tab.
2. Under the *Users* tab, navigate to the desired user.
3. On the right of the row, click the three vertical dots icon, click *Reset password* and press *Send* to send their e-mail address a link to reset their password.

F2.1.4 REMOVING OR DISABLING USERS

When removing a user account, it is recommended that the document in the database belonging to the user is not removed, this allows last modified information for the user to be retained.

1. Open the *Authentication* console tab.
2. In the *Users* tab, navigate to the desired user.
3. On the right of the row, click the three vertical dots icon, click *Disable account* and press *Disable* to disable the account, or click *Delete account* and press *Delete* to delete the account.

F2.1.5 GRANTING AND REVOKING IN-APP ADMINISTRATOR POWERS TO USERS

1. If the UID of the user is not known, follow steps 1 and 4 of F2.1.2 *Adding new users*.
2. On the Firebase console, open the *Database* tab.
3. Under the *Data* tab, click on the “users” collection and select the user document corresponding to the desired UID.
4. If there is no “isAdmin” field, click *Add field*, under *Field* type “isAdmin”, change the *Type* to “boolean”, set *Value* to “true” if administrator access is being granted or “false” if not. Click *Add* to confirm the change.
5. If there is an “isAdmin” field, click it, change *Value* to “true” if administrator access is being granted or “false” if not. Click *Update* to save the change.

F2.1.6 GRANTING USER ACCESS TO WAREHOUSES

1. If the UID of the user is not known, follow steps 1 and 4 of F2.1.2 *Adding new users*.
2. On the Firebase console, open the *Database* tab.
3. Click the “warehouses” collection and take note of (or copy) the IDs of the warehouses to grant the user access to.
4. Under the *Data* tab, click on the “users” collection and select the user document corresponding to the desired UID.
5. If there is no “accessibleWarehouseIDs” field, click *Add field*, under *Field* type “accessibleWarehouseIDs”, change *Type* to “array”, for each warehouse being added, enter its ID under *Value* and click *ADD FIELD* underneath for the next one (ensuring the default *Type* “string” is set for each entry), finally click *Add*.

6. If there is an “accessibleWarehouseIDs” field, on the right of it, for each warehouse being added, click the *Add field* (‘+’) button and set the *Value* to the ID (ensuring the default *Type* “string” is set for each entry) and press *Add*.

F2.1.7 REVOKING USER ACCESS TO WAREHOUSES

1. If the UID of the user is not known, follow steps 1 and 4 of *F2.1.2 Adding new users*.
2. On the Firebase console, open the *Database* tab.
3. Under the *Data* tab, click on the “users” collection and select the user document corresponding to the desired UID.
4. Under the “accessibleWarehouseIDs” drop-down field, delete the warehouse IDs corresponding to the warehouses having their access revoked by pressing the bin icon and confirming by pressing *Delete*.

G DEVELOPER GUIDE

G₁ INTRODUCTION

THIS SECTION ASSUMES PRIOR KNOWLEDGE OF THE USER AND ADMINISTRATION GUIDES (SECTIONS *D HOW TO USE TRAYMASTER* AND *F MAINTENANCE AND ADMINISTRATION*).

This section aims to inform a developer on everything that they would need to know to contribute to the TrayMaster app. It covers an overview of technologies and necessary prerequisite knowledge, details of the current deployment, and other noteworthy information of the existing source code.

G₂ TECHNOLOGIES USED

TrayMaster is a progressive web app (PWA) that is deployed on Heroku. The app itself is programmed in TypeScript React (using webpack). It uses Firebase Cloud Firestore for data storage and Firebase Cloud Functions for some back-end implementations.

The following is a list of technologies used:

- Node.js
- TypeScript
- React
- Sass
- Progressive Web Apps (PWAs)
- Firebase Authentication
- Firebase Cloud Firestore
- Firebase Functions
- GitHub Actions
- Heroku

Standard quick start guides are recommended for familiarising yourself with the above technologies.

G₃ CODE REPOSITORY

THIS SECTION ASSUMES PRIOR KNOWLEDGE OF **GIT** AND **GITHUB**

The source code for TrayMaster can be found at <https://github.com/Kacper-Lubisz/TrayMaster>. The preferred method for modifying the repository is to fork it, although making pull requests to the original repository is an option. To do this, please contact the repository owner to gain Collaborator access to the repository before submitting pull requests.

G4 CONTINUOUS INTEGRATION

THIS SECTION ASSUMES SOME PRIOR KNOWLEDGE OF **GIT** AND **GITHUB ACTIONS**

It is recommended to use GitHub Actions to facilitate Continuous Integration (CI) when developing TrayMaster. This will automatically spot trivial issues and allow Pull Request reviewers to focus on the actual logic instead of other details like code style, syntax, and whether the code will even compile.

The source code already contains a pre-configured Action in the `.github/workflows/nodejs.yml` file. It will run on any Pull Requests into the repository's **staging** and **master** branches. It uses Node.js 12 (latest stable version) on the Windows and Ubuntu operating systems to perform the following:

1. Set up the environment
 - Switch to the relevant branch
 - Set up Node.js
 - Install dependencies
2. Run ESLint
 - This checks against the custom configuration specified in `src/.eslintrc.js`
3. Run Jest unit tests
 - These tests are defined in `.test.ts` and `.test.tsx` files throughout the source code
4. Build the app for deployment
 - This runs the `npm build` command to compile the project for use in production environments
 - It should pick up any compilation issues that other tests have missed

G5 CONTINUOUS DEPLOYMENT (HEROKU)

THIS SECTION ASSUMES SOME PRIOR KNOWLEDGE OF **HEROKU**

It is recommended that the app is hosted on Heroku to ensure a full continuous deployment cycle directly from GitHub. The application can then automatically rebuild itself and redeploy whenever changes are pushed to the **master** branch of your GitHub repository. The application source code already contains configuration files to help you set this up and successfully redeploy your own version on your own domain:

1. Go to dashboard.heroku.com, sign in to a Heroku account, click *New* and then *Create new app*.
2. Enter a name for your deployment – this will also be your chosen domain. For example, you can deploy to `traymaster2.herokuapp.com` by entering "traymaster2" as the app name. Then select a geographical region for the app.
3. To easily build the app, use the [Heroku Buildpack for create-react-app](https://github.com/mars/create-react-app-buildpack) script. To do this, click on the *Settings* tab, then the *Add buildpack* button. Enter `https://github.com/mars/create-react-app-buildpack.git` and click the *Save Changes* button.
4. Click on the *Deploy* tab and select *GitHub (Connect to GitHub)* as your Deployment Method.
5. Click the *Connect to GitHub* button, and authorise Heroku to access your GitHub account.

6. Search for your GitHub repository that contains the application source code, then click the *Connect* button.
7. Choose the **master** branch and click the *Enable Automatic Deploys* button.
8. You must now edit some variables within the source code to support your chosen domain. Open the `.env` file and replace the `PUBLIC_URL` variable (line 2) with your chosen domain. Then open the `static.json` file and replace the `canonical_host` variable (line 2) and `Access-Control-Allow-Origin` headers (lines 7 and 12) with your chosen domain too.
9. Once you push these changes to your GitHub repository's **master** branch, your Heroku app will automatically build itself and deploy to the chosen domain. Click on the *Overview* tab in Heroku to watch the automatically triggered build process and click *View build progress* to identify any build issues.

G6 FIREBASE

THIS SECTION ASSUMES PRIOR KNOWLEDGE OF FIREBASE

TrayMaster uses Firebase Authentication to handle user authentication and Cloud Firestore to handle data storage. These are imported from the respective modules in the `firebase` npm package. The source code that interfaces with Firebase can be found within the `src/core/Firebase/` folder.

G6.1 SETUP

The following step-by-step guide will demonstrate how to set up a new Firebase project to be used with the application.

1. Go to console.firebase.google.com, sign in to a Google account and click *Add Project*.
2. Enter a name for the Firebase project and follow the prompts.
3. Firstly, enable authentication by opening the *Authentication* tab on the left in the *Develop* subheading, open the *Sign-in method* tab, click on the *Email/Password* provider under *Sign-in Providers*, switch it to be enabled and save. If the application is being hosted, add the domain of the host under *Authorized Domains*.
4. Next, set up the database by opening the *Database* tab on the left (under the *Authentication* tab), press *Create database* (in the *Cloud Firestore* section at the top). Follow the prompts (either production mode or test mode can be used as this will be overwritten later). Select an appropriate geographical region as this cannot be changed later, for example `eur2` (europe-west2) for London.
5. An “app” needs to be set up in the Firebase project to provide the application with the URLs and keys it requires to access the Firebase services. To do this, open *Project Overview* (in the top left) and click the ‘</>’ icon under *Get started by adding Firebase to your app*. Give the app a name (such as “TrayMaster”) and continue, some JavaScript is presented with a `firebaseConfig` object containing the required information. To use these in the TrayMaster app, replace the value of each one in the `.env` file in the root directory.
6. Finally, you must set up the database access rules and indices. Using `firebase-tools` (npm install -g firebase-tools), run `firebase deploy --only firestore` (signing in if necessary).

To add warehouses/users, refer to *F2.1.1 Adding new warehouses* and *F2.1.2 Adding new users*.

G6.2 AUTHENTICATION

Firebase Authentication provides secure user sign-in functionality. The application currently requires users to sign in using an email address and password (to add users, see *F2.1.2 Adding new users*). The `Authentication` class in `src/core/Firebase/Authentication.ts` provides this functionality by making calls to the database to authenticate the user and download their settings.

G6.3 DATABASE

Cloud Firestore is a NoSQL database which uses a collection-document tree model. For context, the root consists of a set of collections, where each collection consists of a set of documents. A document itself can contain a set of collections and a set of fields (where each may be a text string, integer, floating-point number, boolean, byte string, map, or document/collection reference). Every collection or document must have an ID (which needs to be unique within its scope). Collection names (IDs) are usually pre-defined and document IDs are usually randomly assigned.

The application uses the database for two main purposes: storing user settings/warehouse access information and storing warehouse data. Every document in the database is modelled in the application as a `Fields` interface which is generically loaded to or saved from in the `Database` class in `src/core/Firebase/Database.ts`. `DatabaseCollection` and `DatabaseObject` (stored in `src/core/Firebase/`) are abstract base classes which represent objects that can be loaded from and saved to the database using the `load` and `stage` (with the `commit` parameter set to `true`) methods.

G6.4 FUNCTIONS

A public HTTP cloud function is provided to output a CSV file containing all of the trays within the warehouse. This function uses the `firebase-tools` module and is located in `functions/src/index.ts`.

G6.5 HOSTING

Firebase hosting can be used but it does not support continuous deployment from GitHub. As such, manual deployments must be performed when new releases are built (done with `npm build`). Some items have been configured, see firebase.google.com/docs/hosting/quickstart in order to set up with a specific Firebase project (setting the static files folder to `build/`). Once complete, the `PUBLIC_URL` in the environment variables (`.env` file) needs to be changed to the URL being deployed to. Use `firebase deploy` on the command line to deploy the build.

G7 WAREHOUSE MODEL

The warehouse model (Warehouse, Zone, Bay, Shelf, Column, Tray) acts as the main interface between the application front-end and the data within the database. Its source code can be found in the `src/core/WarehouseModel/` folder.

Due to the naturally recursive nature of the layers in a warehouse, an abstract warehouse model is used to reduce the amount of duplicated code between each layer. The relevant files can be found in the `src/core/WarehouseModel/LayerStructure/` folder.

This folder contains an abstract base `Layer` class (which extends the `DatabaseObject` class). It is then from this `Layer` class that the `TopLayer`, `MiddleLayer` and `BottomLayer` classes are extended:

- `TopLayer` represents the top layer in the model (no parent, but has children)
 - The `Warehouse` class extends from `TopLayer`
- `MiddleLayer` represents the middle layers in the model (has parent, and has children)
 - The `Zone`, `Bay`, `Shelf`, and `Column` classes extend from `MiddleLayer`
- `BottomLayer` represents the bottom layer in the model (has parent, but no children)
 - The `Tray` class extends from `BottomLayer`

Some algebraic types are also provided to assist in managing these classes:

- The `Layers` type includes `TopLayer`, `MiddleLayer` and `BottomLayer`
- The `UpperLayer` type includes `TopLayer` and `MiddleLayer`
- The `LowerLayer` type includes `MiddleLayer` and `BottomLayer`

The standard warehouse model consists of the classes being used by the front-end to display and manipulate the warehouse, these are stored in `src/core/WarehouseModel/Layers` however, as each class is stored in a separate file, they can be imported neatly from `src/core/WarehouseModel.ts`. Each `UpperLayer` contains a `load` and `stage` method which can specify a minimum layer to load or stage down to.

G8 REACT

THIS SECTION ASSUMES PRIOR KNOWLEDGE OF **REACT**

The React framework is used extensively throughout TrayMaster's source code.

The app uses `react-router-dom` to allow for navigating the app using window history; this is implemented in `src/core/App.tsx`. The `App` component manages most persistent state throughout the app and is the best entry point to start understanding how the source code for different components interacts.

Notably, a lot of state which is related to the warehouse model and authentication is stored in the warehouse model statically.