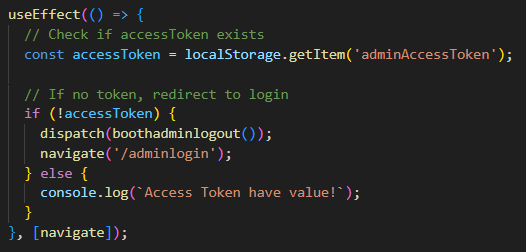
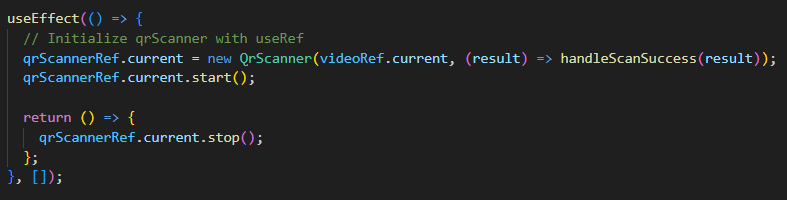
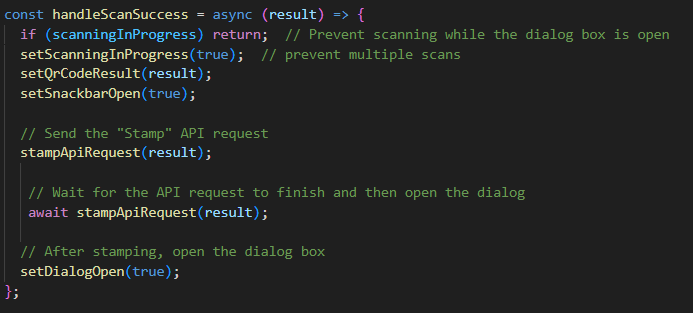
**Qrcodescanner.jsx:**



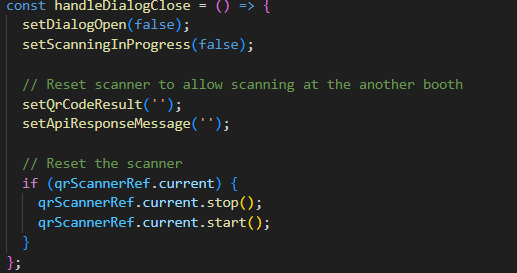
Ensures only authenticated users can access this QR scanner.



Starts the QR scanner when the component loads and it stops when component is removed.

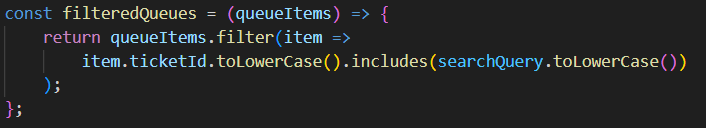


This handles the actions after the QR code has successfully been scanned.

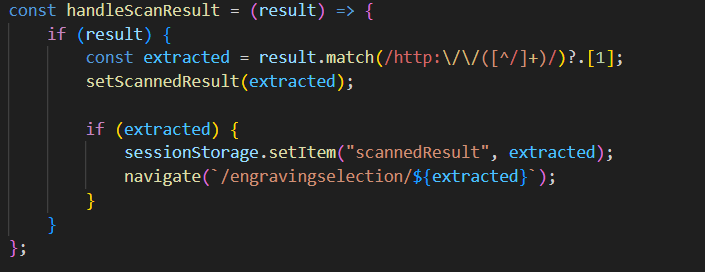


This resets the scanner and state when the dialog is closed. Closes the dialog box and resets scanning-related states (qrCodeResult, apiResponseMessage). And restarts the scanner for the next scan.

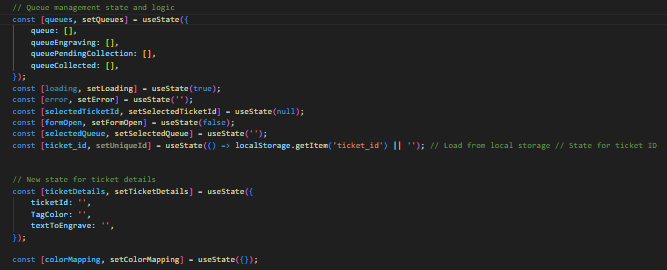
**AdminQueue.jsx:**



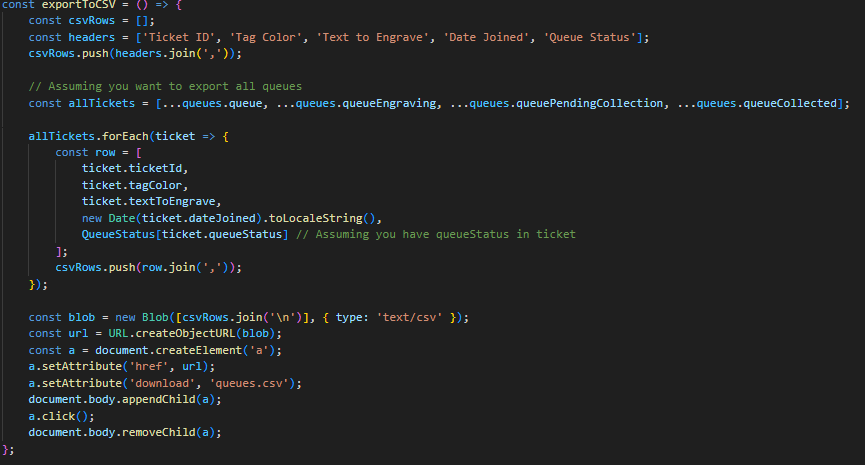
It filters the queue items based on the searchQuery. It enables search functionality by returning only the queue items that match the search input.



This handles the results from the QR scanner. It extracts the userID from the scanned result and navigates it to the engraving selection (EngravingSelection.jsx) page.



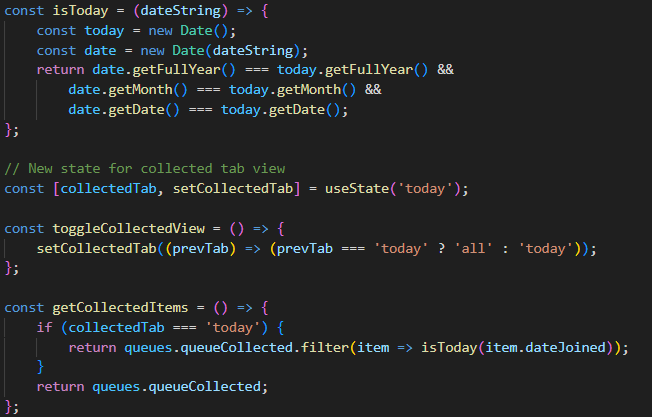
useState is to manage various states in the component, such as UI visibility, ticket details, queue statuses and form handling.



Exports all tickets in the queue to a CSV file, and allows admin with an option to download the ticket data for external use.



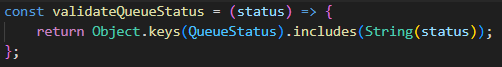
useEffect is to fetch the colour data from an external API, process the data and then store it in a state variable (colorMapping) for use in the component.



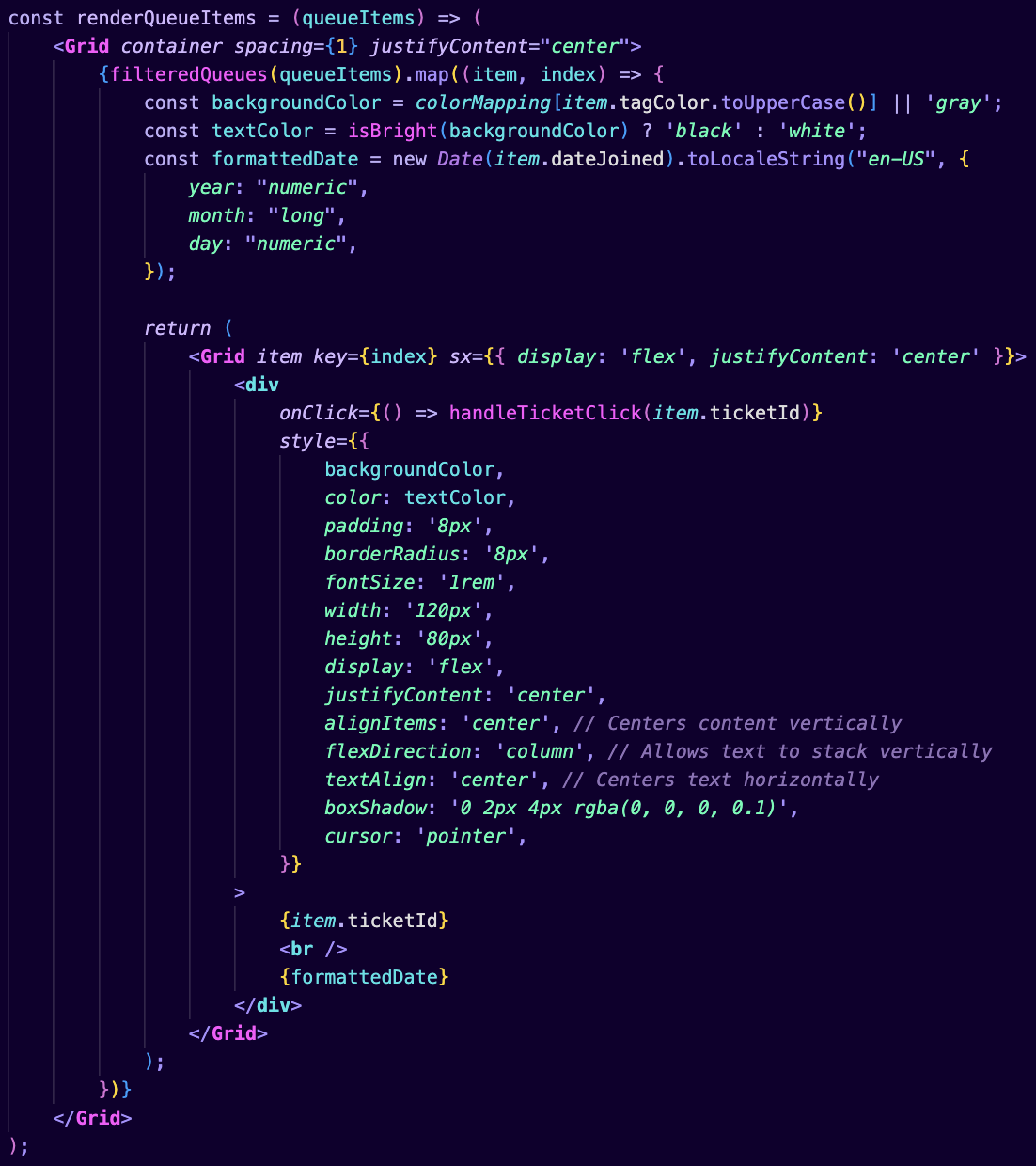
**isToday**: checks if a given date matches today’s date. **toggleCollectedView**: toggles between “Today” and “All” views for the Collected queue. **getCollectedItems**: retrieve queue items for the currently selected view (Today/ All)



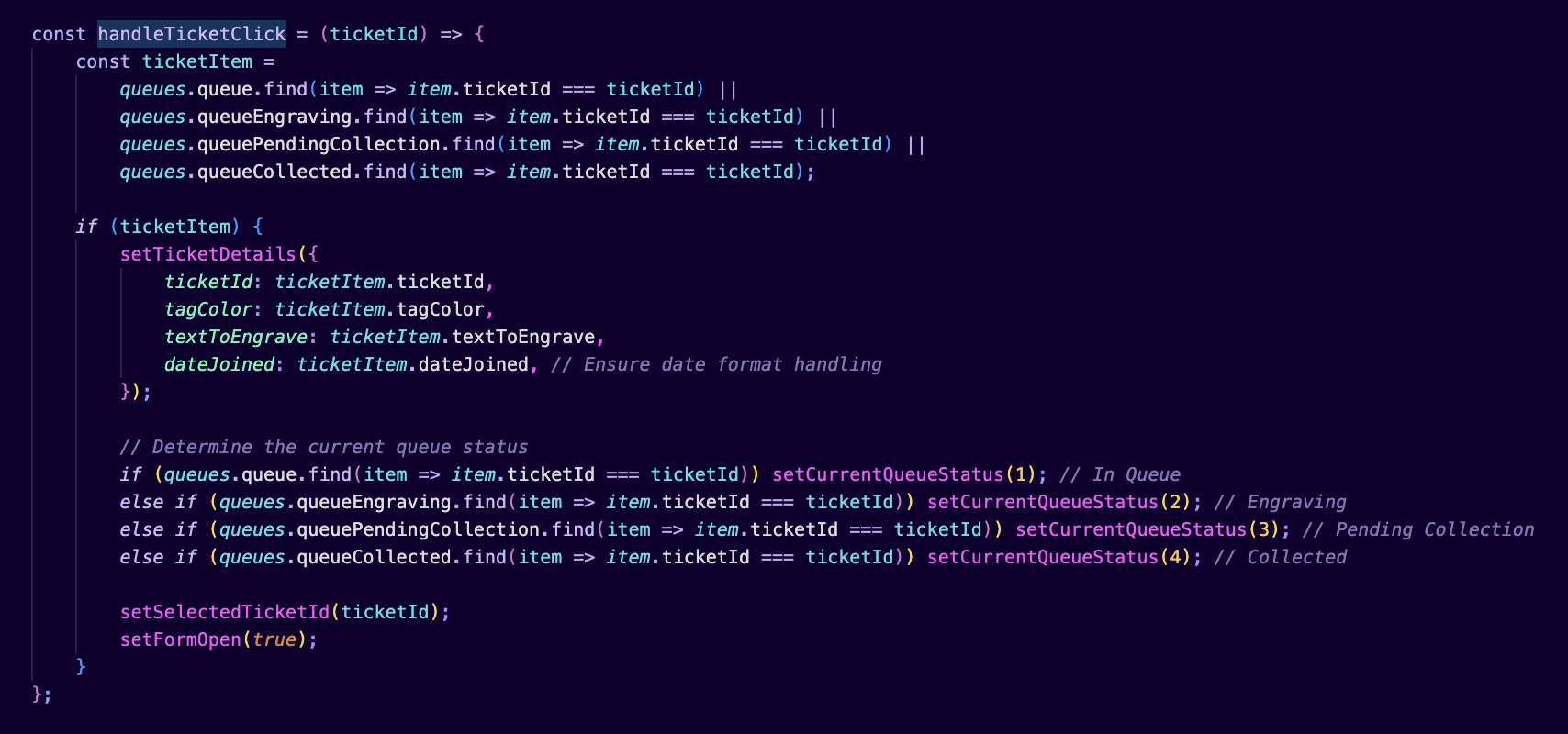
Updates the selectedQueue state with the new queue status.



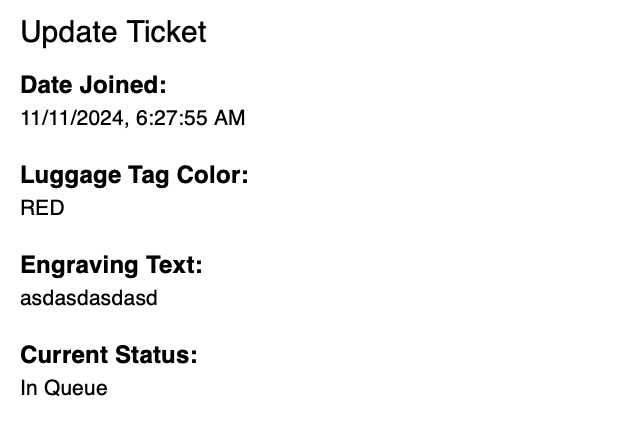
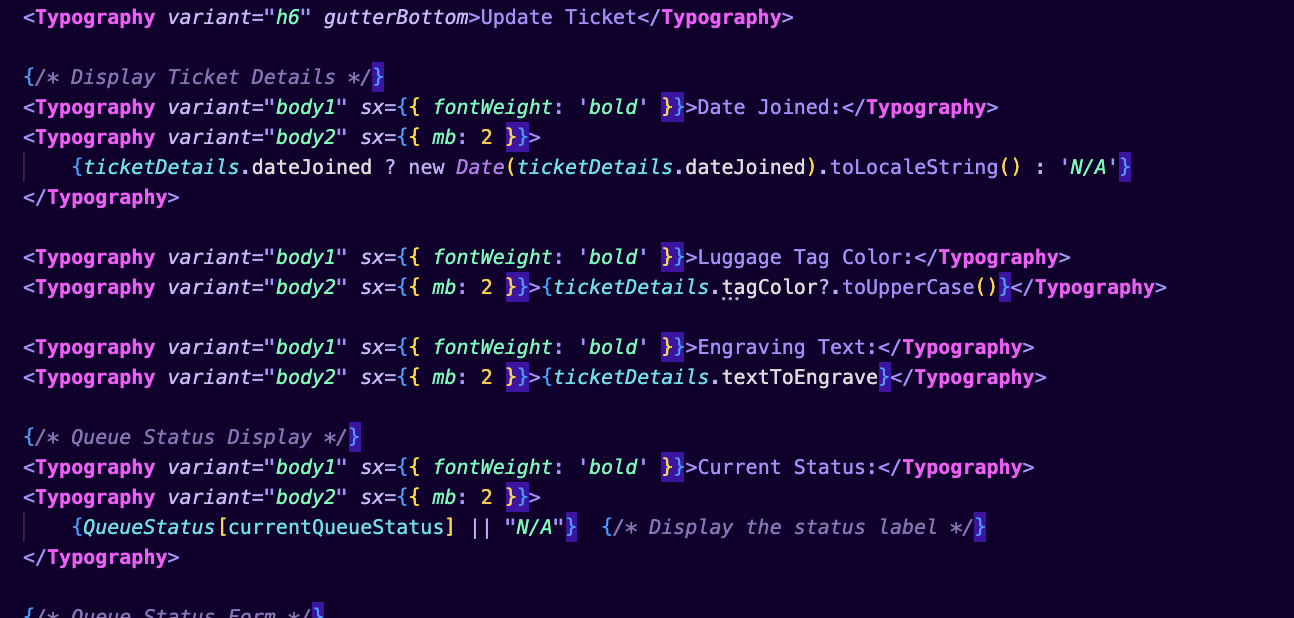
Validates if a queue status is valid based on the QueueStatus object. Ensures only valid statuses are allowed when updating ticket details (queue status).



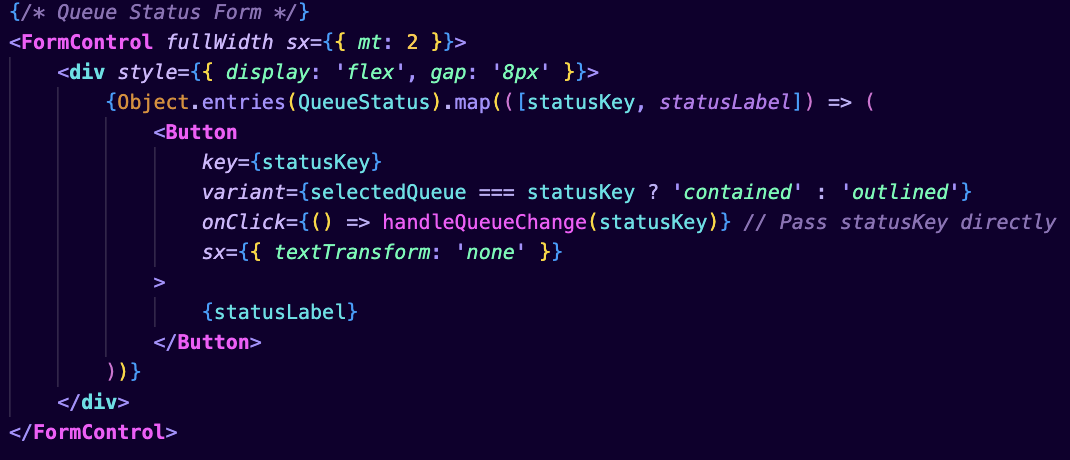
Allows each ticketId in the queue to be uniquely rendered into the queue by defining it by its own ticketId and the background being the colour chosen for the visitor.

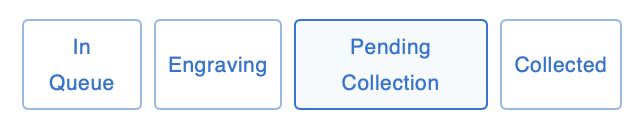


Upon clicking on a certain ticketId the details regarding the ticketId like the colour, textToEngrave and the dateJoined will be retrieved and shown in a modal form for the admin to do any updating of the queue which will also be retrieved



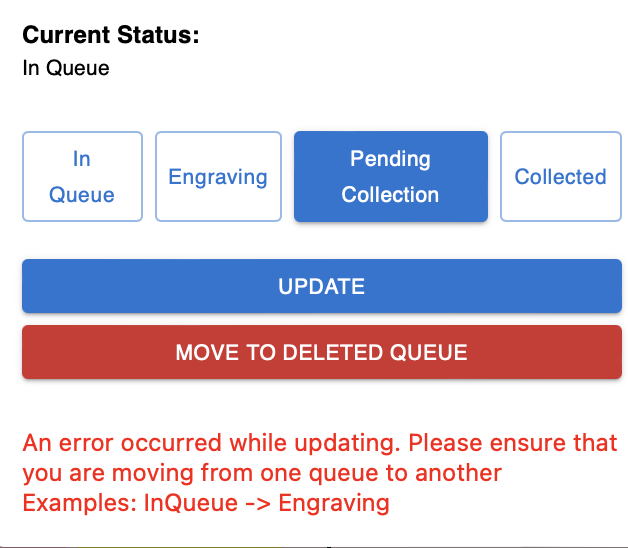
Afterwards the ticketId can be updated to the different queues based off where it is



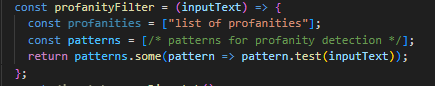
This shows all the queues that the ticketId can be moved to



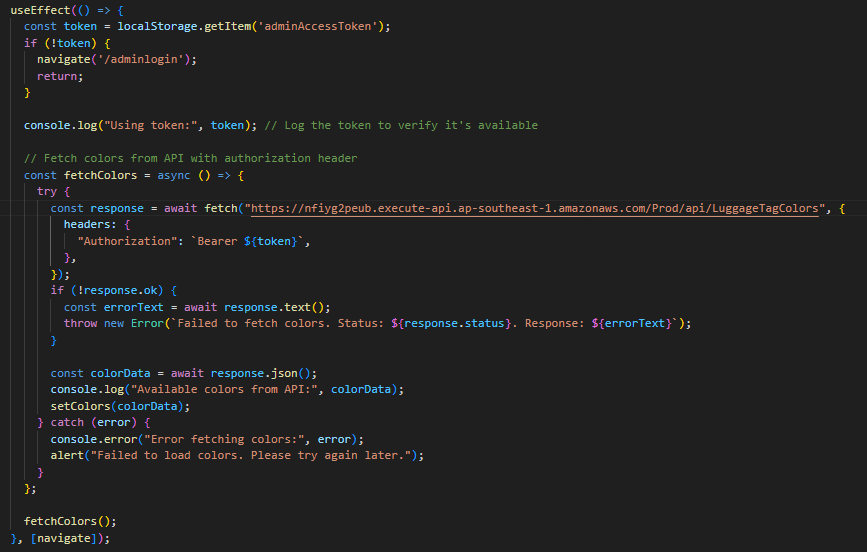
This handles the updating and the movement from one queue to another where an error message will be shown for the various errors



**EngravingSelection.jsx:**



This checks the inputted text before submission for any inappropriate words or patterns and flags them.

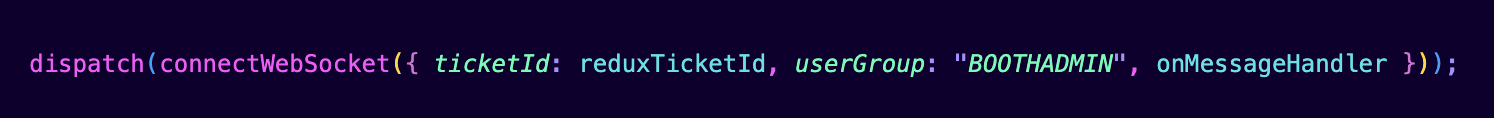


This helps to check if the user is authenticated and also fetches the available colours from the API.

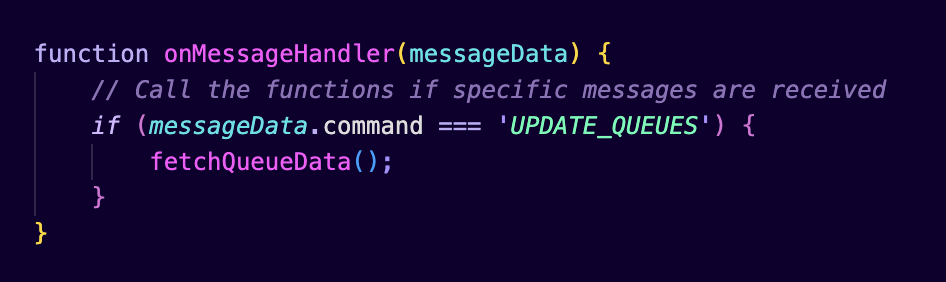


Handles engraving submission by validating the form, preparing the data and sending it to the server.

**WebSocket Lambda:**



Connects to the websocket with the ticketId stored in the Localstorage under the userGroup BOOTHADMIN to receive any commands under BOOTHADMIN



When the command “UPDATE\_QUEUES” is received the function fetchQueueData will be ran again which allows the queue to be constantly updated at each movement or change to the queues.