**FTMS UI - Known Issues**

**Issues**

Cesium Icon Blackout

We found at Ingenuity that once the system had been running for approximately 30mins, all the track icons being displayed on the map would turn black. Reclassifying the tracks, therefore changing the icons, does not recolour them – as such it is almost definitely a problem with Cesium rather than the icon generation library.

Track Replay

Using the webapp’s track replay module may flood and therefore freeze the server if too many track updates are requested at once (depending on the fast forward speed and the number of tracks each x seconds). This means that the server will also be unable to serve webpages quickly to connecting clients whilst flooded.

Web server congestion

Following on from the previous point, we probably should have made two separate servers – one to consume data from Kafka, de-serialise it with Protobuf and send it to web clients, and another simple web server to server webpages and handle authentication/database functions. Currently, these two are combined into the single web server in ‘web\_server’.

Icon sizing is changing resolution instead of scale

The icon sizing setting is currently working by changing the resolution at which the icons are generated at – this is obviously not ideal and was an oversight at the time. It should instead change the Cesium entity’s billboard.scale property, and icon resolution should be kept at a consistent ~64p or 128p.

Manual tracks not saving anywhere

Manual tracks are currently only shown to the person who placed them and any clients connected at the time that they were placed – they disappear to the user if the webpage is refreshed.

Track update stream has not been tested

The original idea was to send track updates along ‘tdn-ui-changes’, however this wasn’t sorted out in time for the end of the sprint, so I made a work-around which stored changes on a local database and used these entries to override any conflicting data on incoming tracks. Relevant lines/file is app.js:163-201.

CSS loading issue

To get colourblind mode working, we used a dodgy trick that loads the CSS files after the settings retrieval callback has been called. This ends up in the CSS files coming into effect after the HTML has been loaded, which may push elements very slightly off screen, which then has the browser generate vertical/horizontal scroll bars depending on the screen/setup. While the functionality is fine, it is not aesthetically pleasing and generally bad practice.

**Bad Practice**

Track table styling is relying on the exact spelling of the affiliation properties

Currently, the class of each row in the track table, which provides its colour and styling, is assigned by concatenating its affiliation property and ‘\_data’. This is case sensitive (as we didn’t use .toLowerCase() but probably should have) and will likely cause problems if the affiliation properties are not all lower case or if the values are changed. Relevant files/lines are track\_table.js:75 & main.css:32-78.

Role management

I believe the login & authentication service is fine, however the role-based part is not implemented well. I currently have an individual folder directory for each role that requires 3 different files each: a html file to list the javascript files theyre allowed to access, a FTMS\_UI class implementation that only loads the modules theyre allowed to access, and a window\_manager.js file that loads their modules in the recommended layout. This is incredibly bad for scalability and maintainability – I only realised recently that I should have used the web server’s view rendering to read user properties, generate the correct html & class script, and load the window manager config from Mongo for a smarter solution.

Layout loading

The window manager we’re using doesn’t provide an update/refresh function, so it is hard to load layouts once it has already been initialised and displayed. Our work-around was to completely reload the client’s UI without refreshing the page, but loading in the new layout config when the layout manager is first initialised.

No weapon request manager

The current interactions between the weapon request module, the auth approval module and the firing module do not have a manager to better define the interactions – while it appears to be working fine without one, the code is quite messy and difficult to understand in some places.

Alerts protobuf not being used

As the EventReplay tool did not send alerts until a later update, we had been testing alerts with raw jsons, and only have untested, commented-out code to de-serialise them if needed. Relevant snippet: app.js:122.