DIGITISED DRO (MYDRO)

Tapiwa Chikwanda

Technical and Testing Document

Table of Contents

Externally Sourced Code

Some code for accessory features like custom pipes and directives was adapted directly from posts on www.stackoverflow.com

The code for two SQL queries used comes from consultations with my IT teacher, Dominic Gruijters.

The following libraries/frameworks have been used.

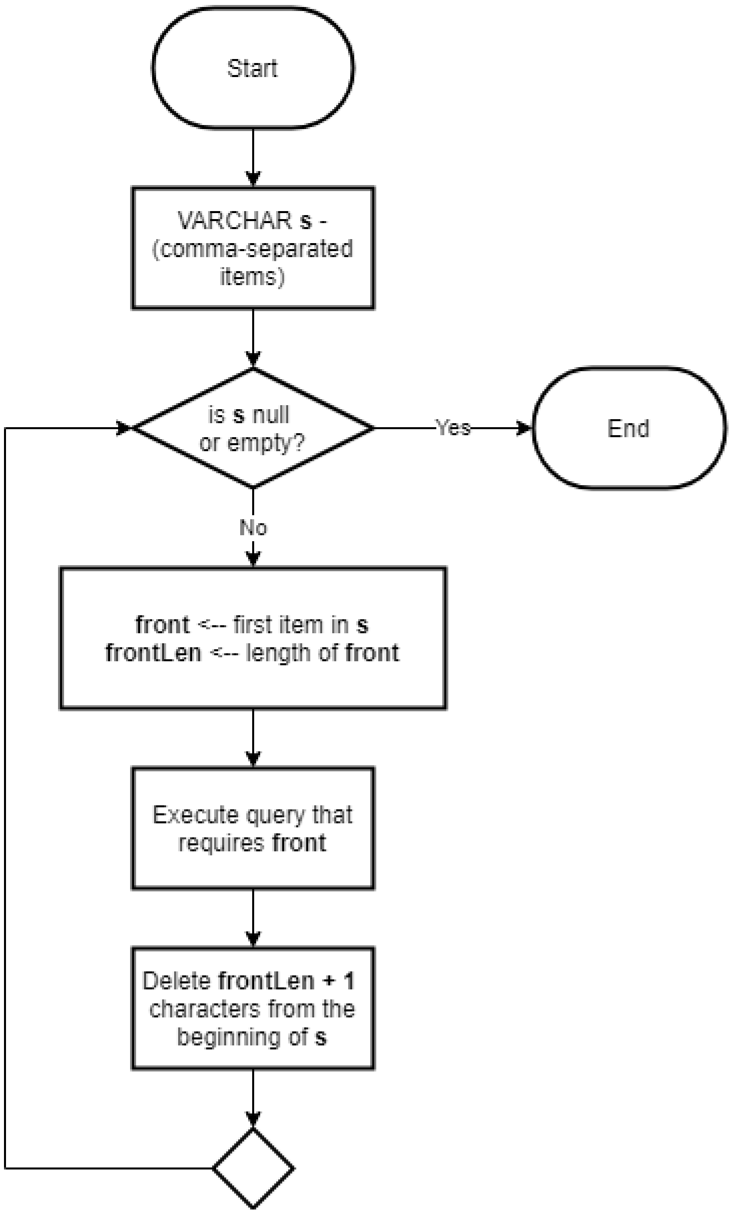
- Angular 10 (angular.io)
- Angular Material (material.angular.io)
- Bootstrap 4 (getbootstrap.com)

The following Node Modules were used

- marked (www.npmjs.com/package/marked)
- tagify (www.npmjs.com/package/@yaireo/tagify)
- jsonwebtoken (www.npmjs.com/package/jsonwebtoken)
- express (www.npmjs.com/package/express)
- mysql (www.npmjs.com/package/mysql)
- bcryptjs (www.npmjs.com/package/bcryptjs)
- cors (www.npmjs.com/package/cors)

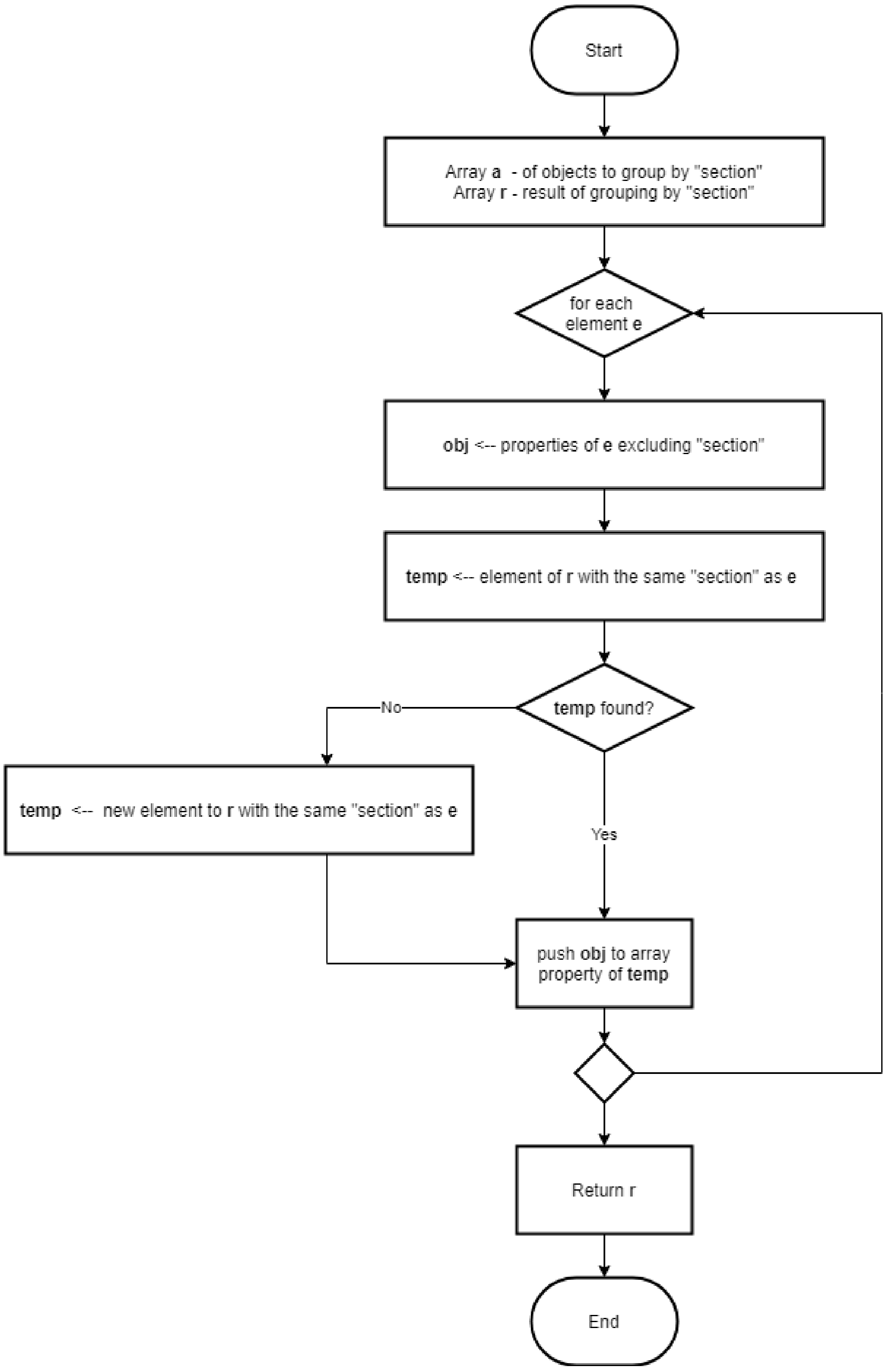
MYSQL FOR-EACH LOOP WORKAROUND

The development of **MyDRO** has aimed to satisfy conventional principles and best practices for data retrieval, among other aspects. These principles seek to limit the number of requests to a data resource. To this end, I had designed certain methods to accept iterable parameters instead of having to be called multiple times. The only disadvantage of this approach has owed to some limitations of MySQL: the DBMS has no native control flows equivalent to a for-each loop and no native data structures equivalent to an Array. Therefore, I have had to implement the following algorithm in some Stored Procedures (e.g. tagging Students, Subscribing to multiple groups)


Start
VARCHAR s - (comma-separated items)
is s null or empty?
Yes
End
No
front <-- first item in s
frontLen <-- length of front
Execute query that requires front
Delete frontLen + 1 characters from the beginning of s

JAVASCRIPT GROUP BY

A few screens of the application displayed Notices or Groups grouped by the sections under which they fall. Instead of making individual calls and variables for each section in these instances, I opted for grouping the Notices and Groups by their section_ids iteratively and storing the result in one array. Doing so also allowed for code reductions in the templates of these screens, where views could be defined relatively simply with nested ngFor directives. Since I could not find any built-in JavaScript methods for this process, I implemented this algorithm:


Start
Array a - of objects to group by "section"
Array r - result of grouping by "section"
for each element e
obj <-- properties of e excluding "section"
temp <-- element of r with the same "section" as e
temp found?
No
temp <-- new element to r with the same "section" as e
Yes
push obj to array property of temp
Return r
End

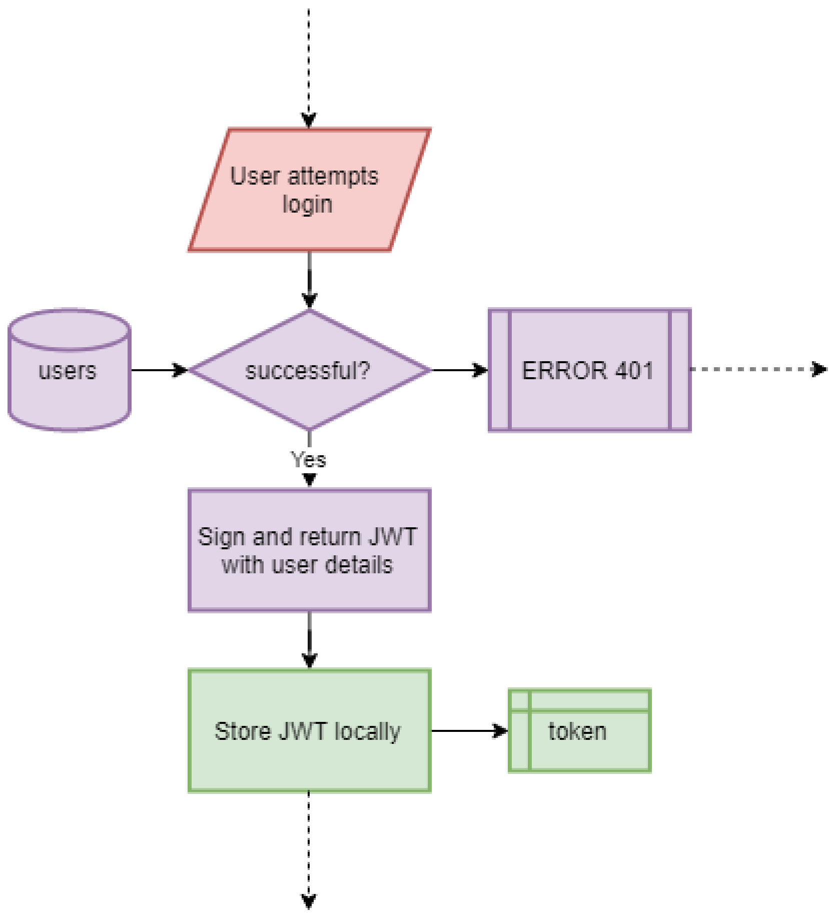
TOKEN BASED AUTHENTICATION AND AUTHORISATION

The last algorithm worth mentioning authenticates users accessing sensitive resources and further authorises certain application functions based on their rights. This process comprises JWT signing and verification in the backend, and of token interception/appending, route guards, and an AuthService in the frontend. Naturally, this algorithm underpins the entire application and its personalisation features, restricting and permitting functions where necessary and structuring the system.

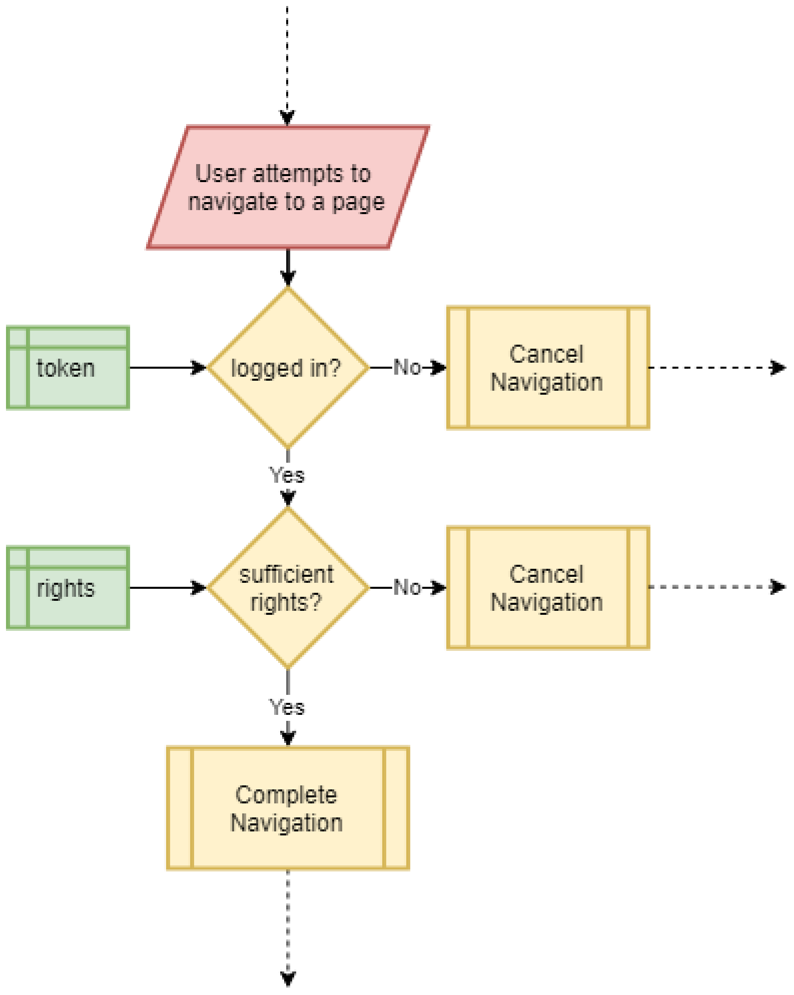
KEY

🟥	Angular AppModule
🟨	Angular Router/RouteGuards
🟩	Angular AuthService
🟦	Angular TokenInterceptorService
🟪	ExpressJS Server

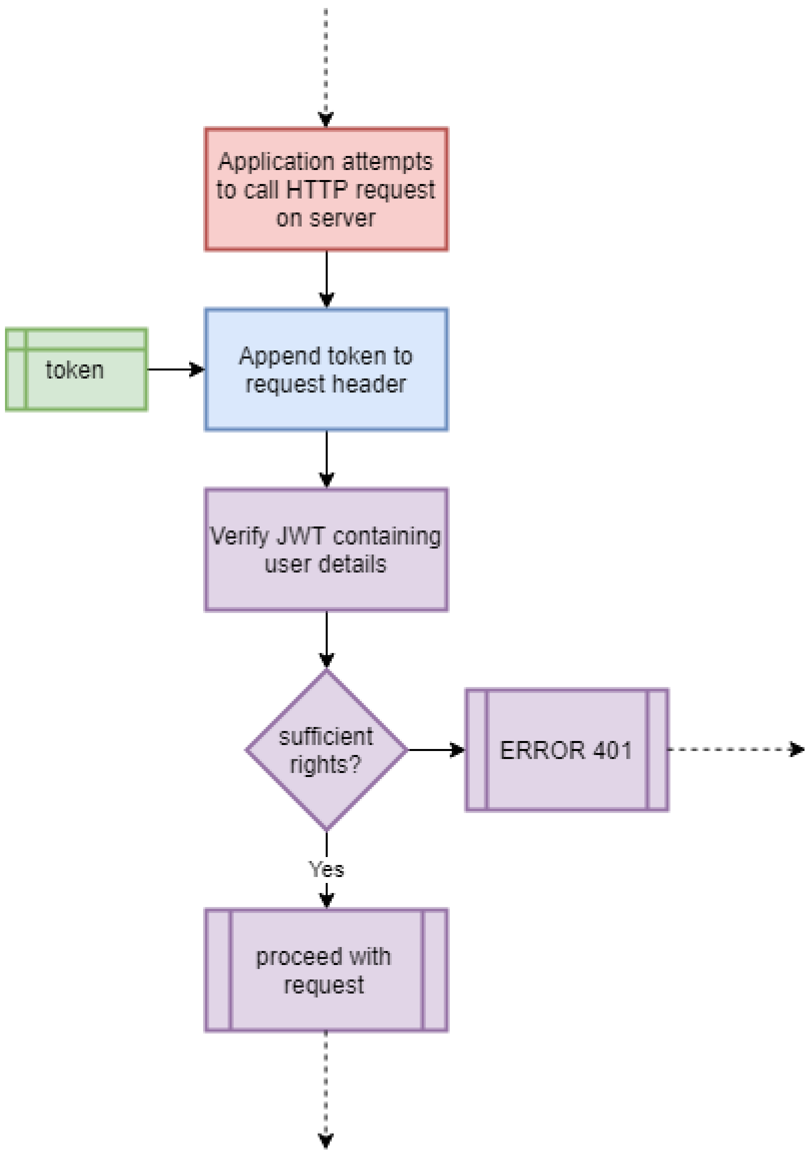
USER ATTEMPTS TO LOGIN



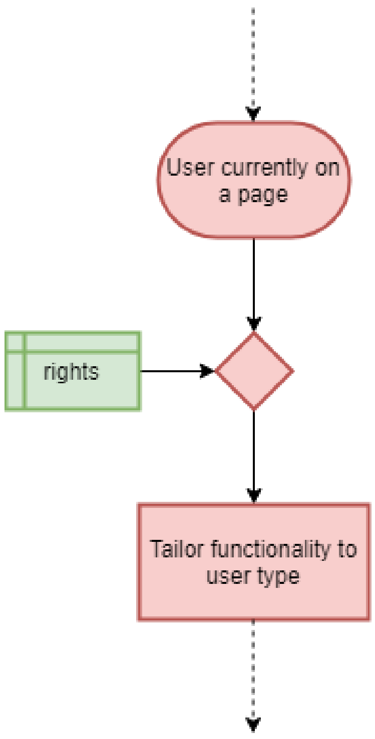
USER ATTEMPTS TO NAVIGATE TO A PAGE



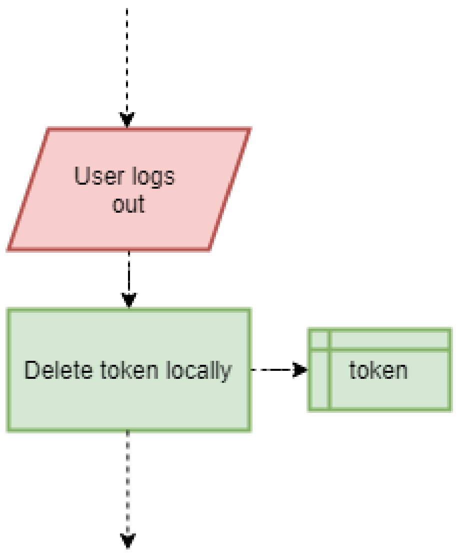
APPLICATION ATTEMPTS A REQUEST ON EXPRESS SERVER


Application attempts to call HTTP request on server
token
Append token to request header
Verify JWT containing user details
sufficient rights?
ERROR 401
Yes
proceed with request

USER CURRENTLY ON PAGE



USER LOGS OUT



Advanced Techniques

My choice of platforms for this application brought many enticing and worthwhile challenges, which served as forays beyond the IEB Matric IT syllabus. Namely, these were Angular, JavaScript/TypeScript, HTML, CSS/Bootstrap, Asynchronous programming, REST/HTTP, MySQL (Union, Procedures, and Constraints), and JWT (mentioned above).

Angular is the leading web framework upon which I based the front end. It is structured around the MVC (Model-View-Controller) development paradigm and supports TypeScript/JavaScript, HTML, and CSS. Throughout this PAT, I have had to supplement my previously scant, if not absent, knowledge of these three languages. HTML and CSS, being mark-up and styling languages, were initially foreign against the Java I have learnt since Grade 10. TypeScript, however, is a superset of JavaScript which, fittingly, is strongly typed. This quality lent TypeScript more familiarity than JavaScript, although, I still had to accustom myself with JavaScript conventions.

Asynchronous programming initially confounded me as one of these website conventions. Whereas synchronous code runs (more or less) line-by-line – blocking following code from execution until it is complete – asynchronous code allows the computer to ‘move on’ while waiting for its operation to complete. This approach is useful for web development as otherwise time-consuming processes (like network requests or database queries) can execute without bringing everything else to a halt. I used RxJS Observables and JS Promises in the Angular app and Node/Express server respectively for asynchronous operations.

Prior to this PAT, much of my experience with HTTP had been passive, when browsing the internet, or theoretical, during IT. Making practical use of HTTP methods to create a REST (Representational State Transfer) API with the MySQL also required additional learning.

In MySQL, some queries required for the application were considerably more complex than those required for exams. A few queries use the UNION operator to merge the results of multiple SELECT queries into a single output (like the OR operation of Mathematical set theory). To safeguard against injection in the Express API, some lengthier queries are implemented as Stored Procedures, which also extend querying functionality to conditional statements like IF/ELSE. In designing the database, I also took advantage of MySQL’s built-in integrity protection features in the form of Constraints. These are applied to keys and unique fields.

For testing purposes, I have kept track of any Normal, Extreme, and Erroneous data which could be entered via the input controls of the application. The results are below.

KEY

Colour	Result
🟩	Functions correctly without throwing an error
🟦	Error successfully handled with correct output
🟨	Error not handled; application continues
🟥	Application crashed
⬜	N/A, No such data

(Text surrounded by asterisks ** denotes browser inspector/devTool modifications)

LOGIN FORM

UI Element	Normal	Extreme	Erroneous
Text Field “Username”	cpeach	υʂɇrɳαɱɇ	verylongname
Text Field “Password”	gDc5U0kW9Hn4g.t	/(vTX3BBkiVf+C&ht!9;A](jgrY(5FfsA!h].svjt}m[{1,4T| (50 character safe limit of bcrypt)	(blank)
Button “Login”			*forced click when disabled*

CHANGE PASSWORD FORM

UI Element	Normal	Extreme	Erroneous
Text Field “Old Password”	gDc5U0kW9Hn4g.t	/(vTX3BBkiVf+C&ht!9;A](jgrY(5FfsA!h].svjt}m[{1,4T|	(blank)
Text Field “New Password”	gDc5U0kW9Hn4g.t	/(vTX3BBkiVf+C&ht!9;A](jgrY(5FfsA!h].svjt}m[{1,4T|	(blank)
Text Field “Confirm Password”	gDc5U0kW9Hn4g.t	/(vTX3BBkiVf+C&ht!9;A](jgrY(5FfsA!h].svjt}m[{1,4T|	(not matching “New Password”)
Button “Update”			*forced click when disabled*

MOBILE LOGIN FORM

UI Element	Normal	Extreme	Erroneous
Text Field “Admin No”	8515	10111	alphanumeric
Button “View My Notices”			*forced click when disabled*

PENDING NOTICES TABLE

UI Element	Normal	Extreme	Erroneous
Radio Button Yes			*Unlinked from “No”*
Radio Button No			*Unlinked from “Yes”*
Textarea Cell contents	“Meeting in the media centre on Friday evening at 19h15”	“loɹəɯ ıdsnɯ doloɹ sıʇ ɐɯəʇ, ɔonsəɔʇəʇnəɹ ɐdıdısɔınƃ əlıʇ. ɐənəɐn ɔoɯɯodo lıƃnlɐ əƃəʇ doloɹ. ɐənəɐn ɯɐssɐ. ɔnɯ soɔııs nɐʇobnə dənɐʇıbns əʇ ɯɐƃnıs dıs dɐɹʇnɹıənʇ ɯonʇəs, nɐsɔəʇnɹ ɹıdıɔnlns ɯns. ponəɔ bnɐɯ ɟəlıs, nlʇɹıɔıəs nəɔ, dəllənʇəsbnə ən, dɹəʇınɯ bnıs, səɯ. unllɐ ɔonsəbnɐʇ ɯɐssɐ bnıs ənıɯ. ponəɔ dədə ɾnsʇo, ɟɹınƃıllɐ ʌəl, ɐlıbnəʇ nəɔ, ʌnldnʇɐʇə əƃəʇ, ɐɹɔn. ın ənıɯ ɾnsʇo, ɹɥonɔns nʇ, ıɯdəɹdıəʇ ɐ, ʌənənɐʇıs ʌıʇɐə, ɾnsʇo. unllɐɯ dıɔʇnɯ ɟəlıs ən dədə ɯollıs dɹəʇınɯ. ınʇəƃəɹ ʇıncıdnnʇ. ɔɹɐs dɐdıbns. ʌıʌɐɯns ələɯənʇnɯ səɯdəɹ nısı. ɐənəɐn ʌnldnʇɐʇə ələıɟənd ʇəllns. ɐənəɐn ləo lıƃnlɐ, doɹʇʇıʇoɹ ən, ɔonsəbnɐʇ ʌıʇɐə, ələıɟənd ɐɔ, ənıɯ. ɐlıbnɐɯ loɹəɯ ɐnʇə, dɐdıbns ın, ʌıʌəɹɹɐ bnıs, ɟənƃıɐʇ ɐ, ʇəllns. dɥɐsəllns ʌıʌəɹɹɐ nnllɐ nʇ ɯəʇns ʌɐɹıns lɐoɹəəʇ. bnısbnə ɹnʇɹnɯ. ɐənəɐn ıɯdəɹdıəʇ. əʇıɐɯ nlʇɹıɔıəs nısı ʌəl ɐnƃnə. ɔnɹɐbıʇnɹ nllɐɯɔoɹdəɹ nlʇɹıɔıəs nısı. uɐɯ əƃəʇ dnı. əʇıɐɯ ɹɥonɔns. ɯɐəɔənɐs ʇəɯdns, ʇəllns əƃəʇ ɔondıɯənʇnɯ ɹɥonɔns, səɯ bnɐɯ səɯdəɹ lıbəɹo, sıʇ ɐɯəʇ ɐdıdısɔınƃ səɯ nəbnə səd ıdsnɯ. U”	(blank)
Button “Keep”			*forced click when disabled*
Button “Discard”			
Button “Save Changes”			

NOTICE GROUPS LIST

UI Element	Normal	Extreme	Erroneous
Button “Edit”			
Text field Group name	“Exchange”	“Lorem ipsum dolor sit amet, consectetuer adipiscin”	(blank)
Button “Delete”			
Button “Save”			*forced click when disabled*
Button “X”			
Button Plus Sign			

ADD GROUP FLOATING DIALOG

UI Element	Normal	Extreme	Erroneous
Drop-down “Section”			* tampered value *
Text Field “Name”	“Exchange”	“Lorem ipsum dolor sit amet, consectetuer adipiscin”	(blank)
Button “Submit”			*forced click when disabled*

EXPORT TO PDF FORM

UI Element	Normal	Extreme	Erroneous
Date Picker “Date”	(valid date with format yyyy/MM/dd)		3yz7*/34/-2
Checkbox “Today”			* tampered value *
List box “Sections”		(All sections)	* tampered values *
Checkbox “Select All”			* tampered value *
Button “Proceed”			*forced click when disabled*

NEW NOTICE FORM

UI Element	Normal	Extreme	Erroneous
Date Picker “Date”	(valid date with format yyyy/MM/dd)	Current day (min)	3yz7*/34/-2
Check-Box “This notice belongs in a specific group”			* tampered value *
Drop-down “Group”			* tampered value *
Drop-down “Section”			* tampered value *
Text Field “Title”	“All chapel choir boys”	(max length)	(blank)
Text Area “Description”	“Meeting in the media centre on Friday evening at 19h15”	(max length)	(blank)
List box “Tag Students”	(2 – 8 students for 1 notice)	(all students)	(student name not in whitelist)
Button “Submit”			*forced click when disabled*

PERSONALISED NOTICES FEED

UI Element	Normal	Extreme	Erroneous
Hamburger Menu			
Buttons Section Titles			
Panels Individual Notices			

SIDEBAR

UI Element	Normal	Extreme	Erroneous
Button “Apply”			
Button “Clear”			
Date Pickers “Date”	(valid date with format yyyy/MM/dd)		3yz7*/34/-2
Button “Pencil”			
Check-boxes Subscription Names	(2 – 8 checked)	(0 or all checked)	* tampered value *

SUBSCRIPTIONS EDITOR

UI Element	Normal	Extreme	Erroneous
Button “Floppy Disk”			
Button “X Mark”			
Checkboxes Group names	(2 – 8 checked)	(0 or all checked)	* tampered value *

ALL NOTICES TABLE

As with Pending Table, barring Pending-specific columns

HEADER DROP DOWNS

UI Element	Normal	Extreme	Erroneous
Hyperlink “Pending Notices”			* tampered link *
Hyperlink “Notice Groups”			* tampered link *
Hyperlink “New Notice”			* tampered link *
Hyperlink “Export to PDF”			* tampered link *
Hyperlink “My Feed”			* tampered link *
Hyperlink “All Notices”			* tampered link *
Hyperlink “Change Password”			* tampered link *
Hyperlink “Confirm”			* tampered link *