

# DIGITISED DRO (MYDRO)

Tapiwa Chikwanda  
System Design Document

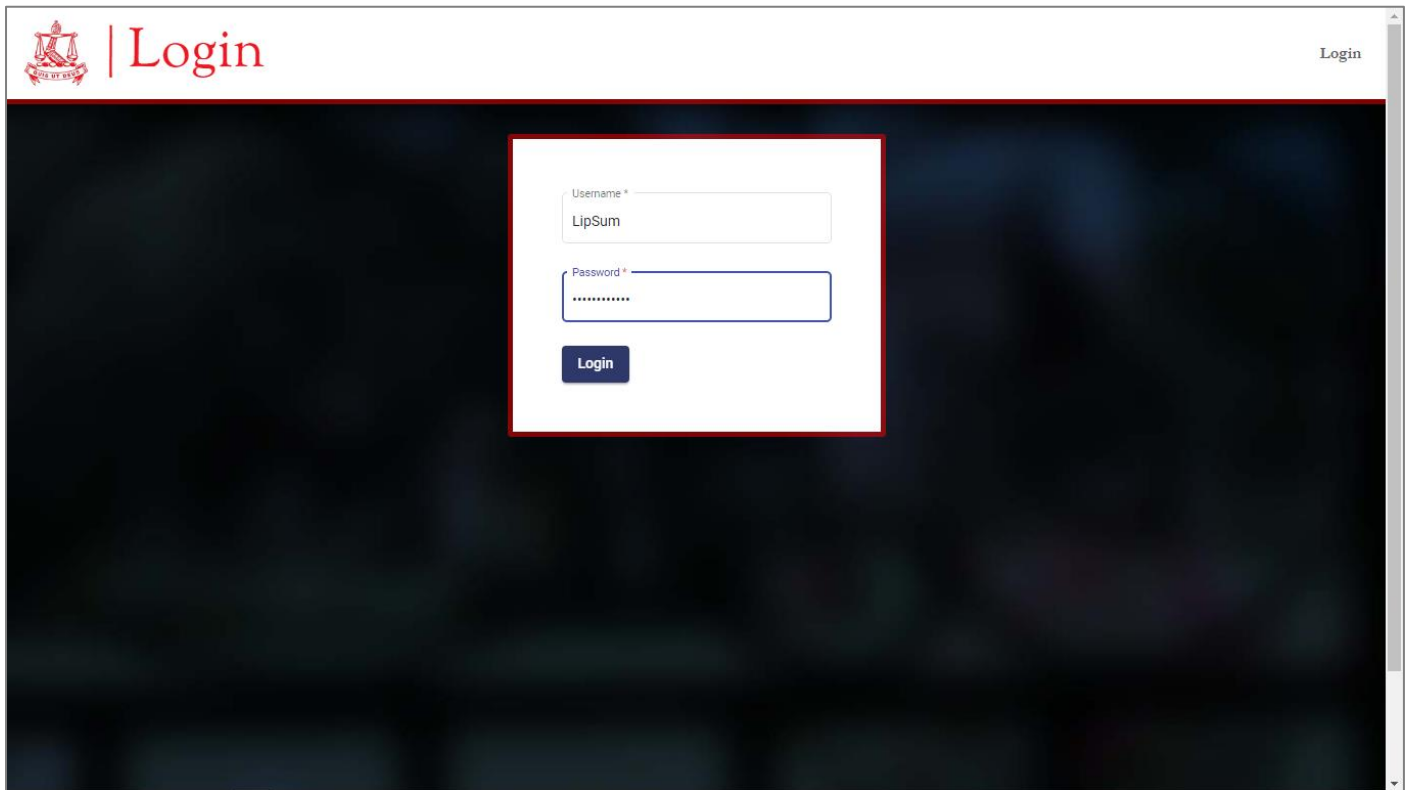
Table of Contents.....	2
User Interface Design.....	4
Desktop.....	4
LogIn Form .....	4
Landing Page (Admin) .....	5
Pending Notices Table (Admin).....	10
Notice Groups List (Admin).....	11
New Notice Form (Staff) .....	13
Export to PDF Form (Staff).....	16
Personalised Notices Feed .....	18
View All Notices Table.....	21
Header Drop Downs.....	22
Change Password Form .....	23
Mobile.....	24
Sign In Form .....	24
Personalised Notices Feed .....	25
Header Menu .....	27
Sequencing.....	28
FlowChart Legend .....	28
Login Page .....	29
Home Page .....	30
Navigation Links .....	30
Pending Page.....	31
Groups Page .....	32
New Page .....	33
Export Page .....	34
All Page.....	35
Feed Page.....	36
Password Page .....	38
Navigation Bar.....	39
Class Design.....	40
Typescript Interfaces .....	40

Overview .....	40
Notice Types.....	41
Group Types .....	43
Section.....	43
User Types.....	44
Backend – Node (Express) REST Server .....	44
MySQL Database .....	44
Express Router .....	46
Frontend – Angular Single Page Application (SPA).....	48
NgModules.....	48
Services .....	49
Guards.....	52
Secondary Storage Design .....	54
Users .....	54
Students .....	55
Notices .....	56
Groupings.....	57
Sections.....	58
Subscriptions.....	59
Mentions.....	60
Overview and Relationships .....	61
Explanation of Secondary Storage Design .....	62

## DESKTOP

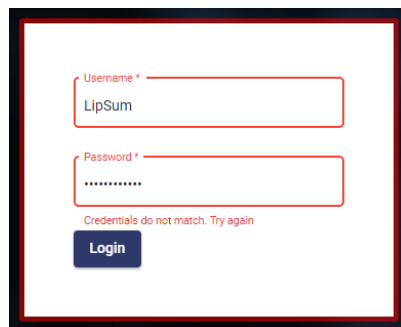
Unless specified otherwise, the screenshots below pertain to Pupil users

## LOGIN FORM



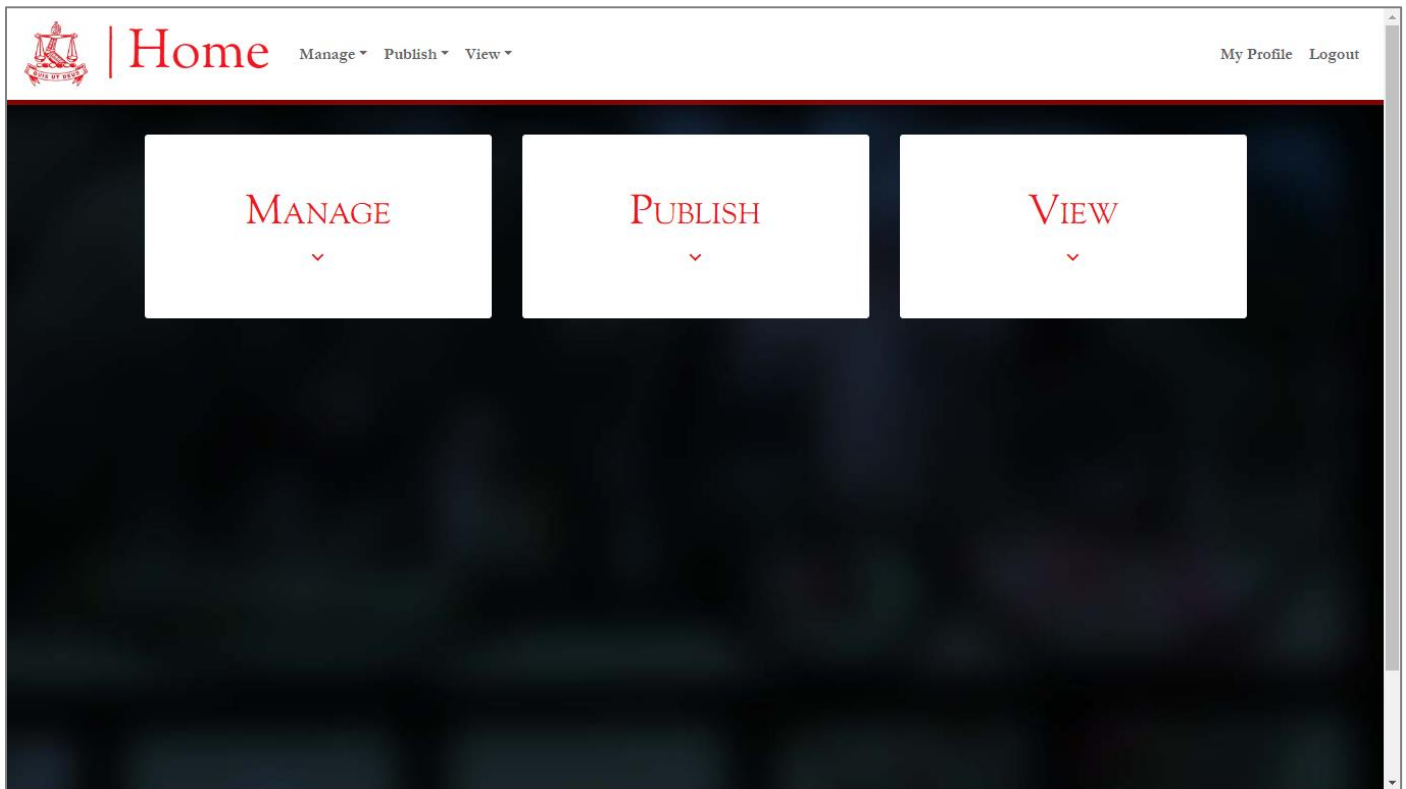
The screenshot shows a desktop login form. At the top left is a logo with a crest and the text 'Login'. At the top right is a 'Login' link. The main form area has a dark background. A white box with a red border contains the login fields: a 'Username' field with the text 'LipSum', a 'Password' field with masked characters, and a 'Login' button.

UI Element	Function
<b>Text Field</b> "Username"	Allows user to enter their username
<b>Text Field</b> "Password"	Allows user to enter their password
<b>Button</b> "Login"	If details valid – directs user to their relevant home page If details invalid – formats the dialog box as below



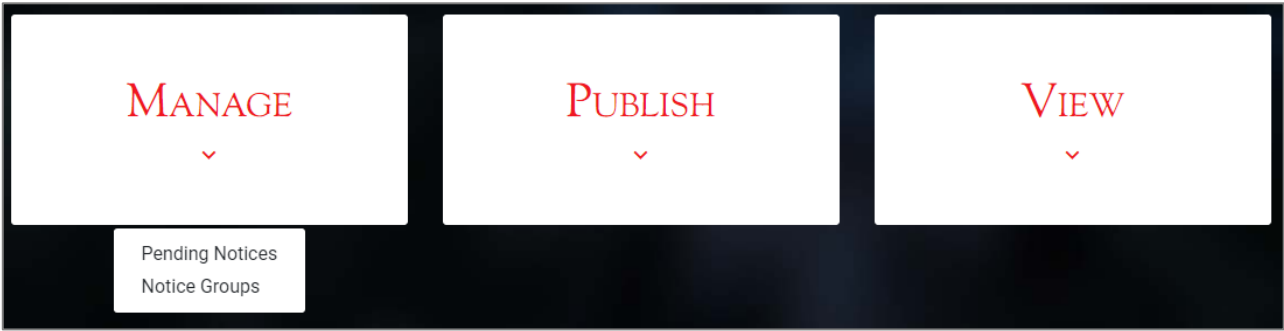
The screenshot shows the login form with an error message. The 'Username' and 'Password' fields are highlighted with red borders. Below the password field, the text 'Credentials do not match. Try again' is displayed. The 'Login' button is still present.

## LANDING PAGE (ADMIN)



UI Element	Function
Card "Manage"	On Click – Reveals drop-down menu of hyperlinks to Admin screens
Card "Publish"	On Click – Reveals drop-down menu of hyperlinks to Staff screens
Card "View"	On Click – Reveals drop-down menu of hyperlinks Student screens

MANAGE DROP-DOWN MENU



UI Element	Function
Hyperlink “Pending Notices”	Directs user to the “Pending” webpage
Hyperlink “Notice Groups”	Directs user to the “Groups” webpage

PUBLISH DROP-DOWN MENU



UI Element	Function
Hyperlink “New Notice”	Directs user to the “New” webpage
Hyperlink “Export to PDF”	Directs user to the “Export” webpage

VIEW DROP-DOWN MENU



UI Element	Function
Hyperlink “My Feed”	Directs user to the “My Feed” webpage
Hyperlink “All Notices”	Directs user to the “View All” webpage



STAFF AESTHETIC VARIATION

“Manage” Card Omitted

<div>PUBLISH</div> <div>▼</div>	<div>VIEW</div> <div>▼</div>
---------------------------------	------------------------------


PUPIL AESTHETIC VARIATION

“Manage” and “Publish” Card Omitted

VIEW

▼

## PENDING NOTICES TABLE (ADMIN)


Pending
Manage ▾ Publish ▾ View ▾
My Profile Logout

Date	Group	Title	Description	User	Approve?
25 Feb	Service	B Block Smile Programme	The following boys who signed up for SMILE classes in the first quarter, please meet at the squash court parking area <b>**tomorrow**</b> afternoon at 14h10: Tristan Baker; Bradley Banda; Ayanda Cele; Thomas Deist; Nicholas Howarth; Benjamin Jarvis; Ross Keep; Matthew Marx; Simon Le Vieux; Siyabonga Ncube; Dilan Patel; Heinrich Reyneke; Sello Stone-Mboweni; Anthony Turner.	A Roddie	<input checked="" type="radio"/> Yes <input type="radio"/> No
25 Feb	Service	B Block Service Representatives	Please meet Mr Coventry in the Media Centre after Chapel <b>**tomorrow**</b> : K Ramnath; C Leisegang; R <b>Fleming</b> ; R le Sueur; B Seeiso; V Getyengana ; J Jordaan; N Manyara; T Deist; J-L Ferreira; N Holt <div>Keep Discard</div>	C Peacher	<input type="radio"/> Yes <input checked="" type="radio"/> No
21 Feb	Investment Club	Investment Club <div>Keep Discard</div>	Meeting in the media centre on Friday evening at 19h15	P Bradley	<input type="radio"/> Yes <input checked="" type="radio"/> No
21 Feb	Snell Society	Snell Society	there is a Snell Society meeting this Sunday at 19h45 in the Centenary Centre. Nathan Bau and Sam Frost will be doing their presentations. Staff and boys are encouraged to attend.	P Bradley	<input type="radio"/> Yes <input type="radio"/> No
25 Feb	Sanatorium	Physio with Leanne St Clair in First Aid Room	14h00 M Christodoulou; 14h30 L van Aardt; 15h00 M Solms; 15h30 S Heinemann; 16h00 A Xulu; 16h30 K Asumaning	G Murrish	<input checked="" type="radio"/> Yes <input type="radio"/> No

Save Changes

UI Element	Location	Function
Radio Button Yes	Column: [Approve?]	Once the page is saved, includes the notice in the DRO and removes it from “Pending”
Radio Button No	Column: [Approve?]	Once the page is saved, excludes the notice from the DRO and removes it from “Pending”
Cell Contents	Columns: [Title]; [Description]	On click – show text area where user can edit the notice before approving
Button “Keep”	Cell Editing	Confirm edits
Button “Discard”	Cell Editing	Revert edits
Button “Save Changes”	Bottom Right	Updates the `notices` Table with any changes

## NOTICE GROUPS LIST (ADMIN)

The screenshot shows the 'Groups' admin interface. At the top, there is a header with a logo, the title 'Groups', and navigation links: 'Manage', 'Publish', and 'View'. On the right, there are links for 'My Profile' and 'Logout'. The main content area is divided into three sections: 'General Notices', 'Sport', and 'Clubs and Societies'. Each section contains a list of groups with 'Edit' and 'Delete' buttons. The 'Lost and Found' group in the 'General Notices' section is highlighted with a blue border. A blue circular button with a white plus sign is visible at the bottom right of the 'Clubs and Societies' section.

Category	Group Name	Edit	Delete
General Notices	Exchange	Edit	Delete
	Journey	Edit	Delete
	Lost and Found	X Save	Delete
	Music	Edit	Delete
	Service	Edit	Delete
	Theatre	Edit	Delete
	Uni	Edit	Delete
Sport	Basketball	Edit	Delete
	Gym Times	Edit	Delete
	Hockey	Edit	Delete
	Squash	Edit	Delete
Clubs and Societies	Chapel Choir	Edit	Delete
	Debating Society	Edit	Delete

UI Element	Function
<b>Button</b> "Edit"	Produces text field where user can edit the name of the notification group
<b>Button</b> "Delete"	Removes the notification group from the system
<b>Text field</b> Group name	Allows user to enter the name of a group
<b>Button</b> "Save"	Updates the `groupings` table with changes
<b>Button</b> "X"	Reverts changes
<b>Button</b> Plus Sign	Opens "Add Group" floating dialog

ADD GROUP FLOATING DIALOG

Unit

General Notices

Sport

Clubs and Societies

Sanatorium

Restrictions

Section \*

Edit Delete

Edit Delete

Edit Delete

Edit Delete

Edit Delete

▼ Name \*

Submit

UI Element	Function
Drop-down "Section"	Allows user to choose from the available notification group sections
Text Field "Name"	Allows user to type a name for the new notification group
Button "Submit"	Inserts the new notification group into the `groupings` table

NEW NOTICE FORM (STAFF)

 | New

Manage ▾ Publish ▾ View ▾

My Profile Logout

DATE

Display Date \*

yyyy/mm/dd

UI Element	Function
Date Picker "Date"	Allows user to select the date on which their notice should display

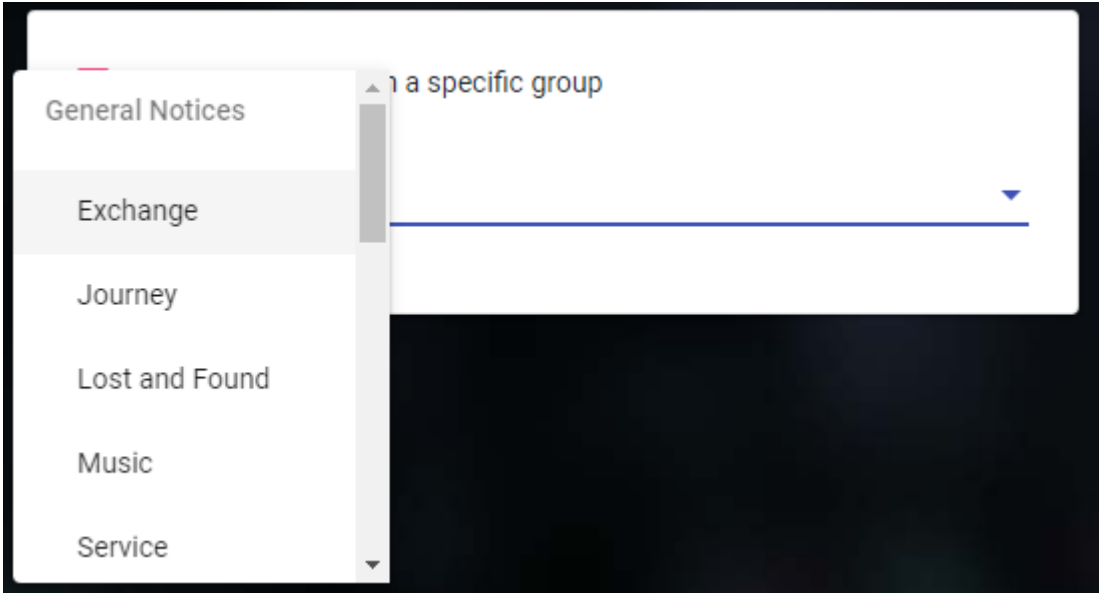
SPECIFIC GROUP

☒ This notice belongs in a specific group

Group \*

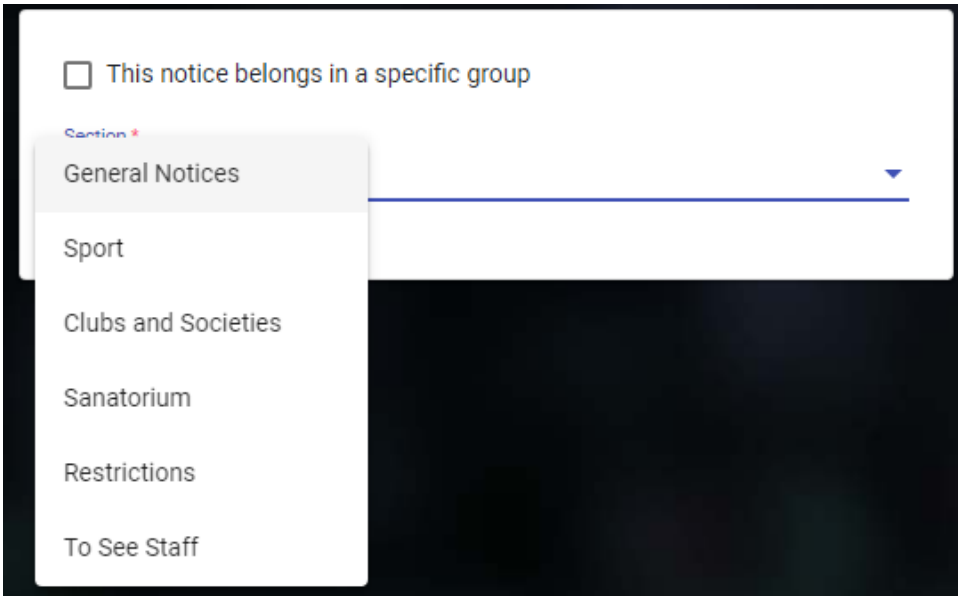
UI Element	Function
Check-Box "This notice belongs in a specific group"	Allows user to specify whether the notice falls under a specific group

GROUP (IF SPECIFIC)



UI Element	Function
Drop-down “Group”	Allows user to select a group under which their notice falls

SECTION (IF NOT SPECIFIC)



UI Element	Function
Drop-down “Section”	Allows user to select a section under which their notice falls

TITLE AND DESCRIPTION

Title \*0 / 100

Description \*0 / 1000

UI Element	Function
Text Field "Title"	Allows user to enter a title for their notice
Text Area "Description"	Allows user to enter a description for their notice

TAG STUDENTS

Tag Students (Optional)

Conor MacColl xba

Tristan Baker

Bradley Banda

UI Element	Function
List box "Tag Students"	Allows user to tag specific students manually

SUBMIT

Submit

UI Element	Function
Button "Submit"	Inserts a tuple with the above information into the `notices` and `mentions` tables

## EXPORT TO PDF FORM (STAFF)

The screenshot shows a web interface for exporting to PDF. At the top, there is a header with a logo, the word 'Export', and navigation links: 'Manage', 'Publish', and 'View'. On the right, there are links for 'My Profile' and 'Logout'. The main content area is a form with a dark background. It features a date picker labeled 'Date:' with a dropdown menu showing '2020/02/25' and a checkbox labeled 'Today'. Below this is a 'Sections:' dropdown menu with a list of sections: 'General Notices', 'Sport', 'Clubs and Societies', 'Sanatorium', 'Restrictions', and 'To See Staff'. There is also a checkbox labeled 'Select All'. A 'Proceed' button is located at the bottom right of the form.

UI Element	Function
<b>Date Picker</b> "Date"	Allows user to select a date whose notices to export
<b>Checkbox</b> "Today"	Sets the date picker to the current date
<b>List box</b> "Sections"	Allows user to select multiple sections to export
<b>Checkbox</b> "Select All"	Selects every available section
<b>Button</b> "Proceed"	Generates PDF



## OUTPUT

Print

Total: 2 sheets of paper

PrintCancel

Destination

Microsoft Print to PDF

Change...

Pages

All

e.g., 1-5, 8, 11-13

Colour

Colour

Options

Simplify page

More settings

Print using system dialogue... (Ctrl+Shift+P)

Daily Routine Order

25 February 2020 : TUESDAY

GENERAL NOTICES

1 University Visits: Who: University of Pretoria When: Today @ 16h30 Where: KLT

2 All B Block Boys: To meet in the KLT today at 14h05

3 Service: Tatham C Block - Esiphetwini: Wednesday 14h10 at the turning circle

West C Block - Notties Prep: Wednesday 14h10 at the turning circle

4 B Block Smile Programme: The following boys who signed up for SMILE classes in the first quarter, please meet at the squash court parking area tomorrow afternoon at 14h10: Tristan Baker, Bradley Banda, Ayanda Cele, Thomas Deist, Nicholas Howarth, Benjamin Jarvis, Ross Keep, Matthew Marx, Simon Le Vieux, Syabonga Ncube, Dilan Patel, Heinrich Reyneke, Sello Stone-Mboweni, Anthony Turner.

5 B Block Service Representatives: Please meet Mr Coventry in the Media Centre after Chapel tomorrow: K Ramnath, C Leisegang, R Fleming, R le Sueur, B Seeiso, V Getyengana ; J Jordaan; N Manyara; T Deist; J-L Ferreira; N Holt

6 Inter House Theatre: Two representatives from each house to come to drama classroom tonight at 19h30 for playwright introduction and inter house competition rules.

SPORT

1 Open Hockey Players: Aitkens astro : Tomorrow 14h45 - 16h00. All open hockey players to attend match sessions for 2nds-5ths.

2 Senior Squash Team vs Hilton College: Away tomorrow - Bus leaves at 14h00. R Ingledew, C Oellermann, M RossO, M Flanagan, N Bedingham, J Irons, A Nduru, B Harper Dress: Blue shirts, white shorts

CLUBS AND SOCIETIES

1 Chapel Choir: Rehearsals and performances for this week are as follows: Monday and Thursday 17h30 to 18h40; Tuesday 17h20 to 17h50 (due to the Pancake Race), Wednesday 07h30 (Full School Eucharist performance), Friday 18h00 to meet in Chapel for warm-up prior to @Bandroom performance.

2 Flying Club: All boys joining the flying club just meet Mr Silk at 14h05 in the music school to discuss this weekend's Flying

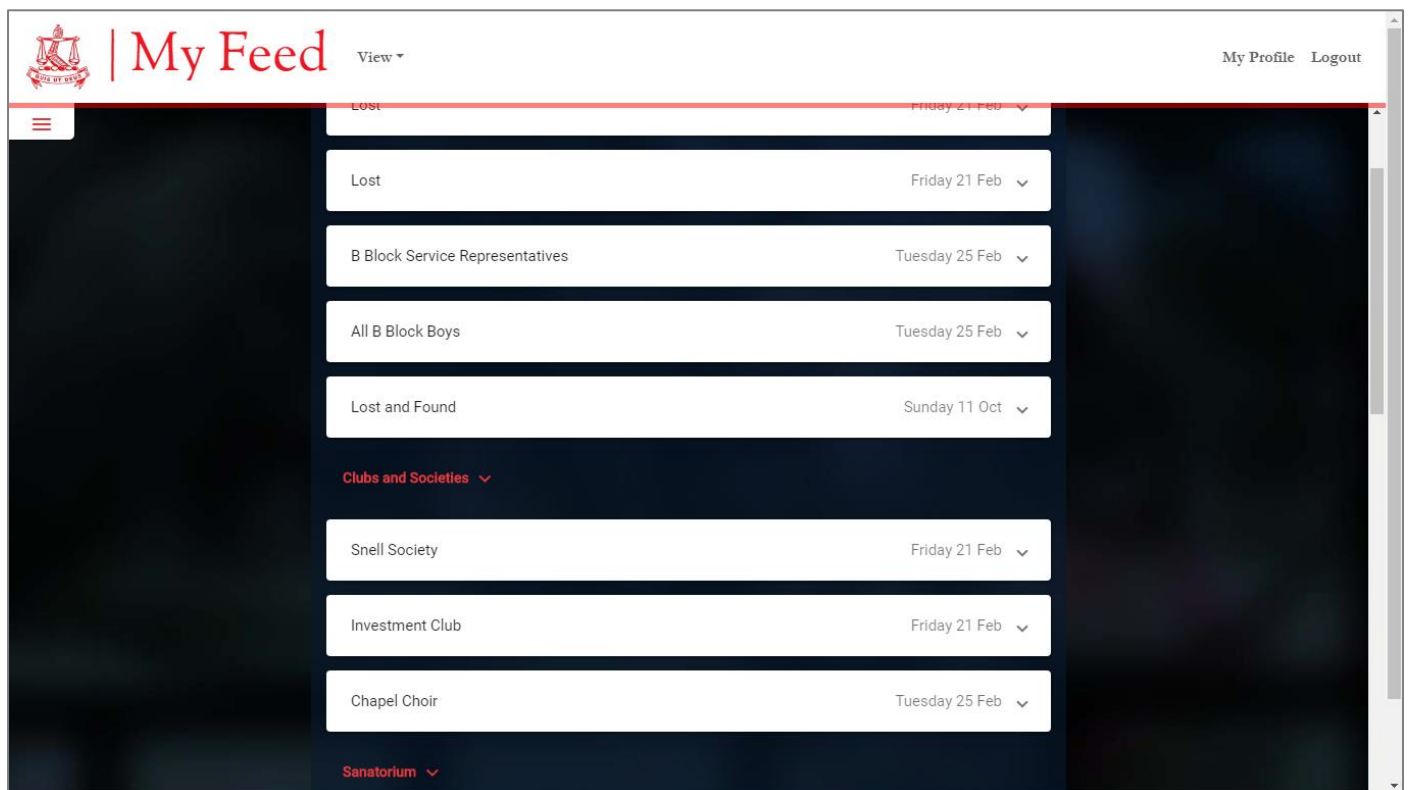
### UI Element

### Function

Browser Print Dialog

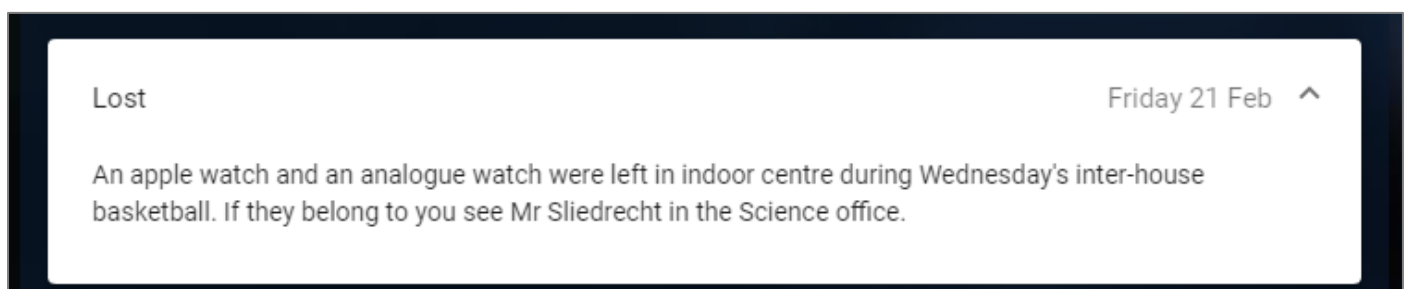
Display and allow download/printing of exported PDF


## PERSONALISED NOTICES FEED



UI Element	Function
<b>Hamburger Menu</b>	Reveals/Conceals the sidebar
<b>Buttons</b> Section Titles	Allows user to expand and collapse specific sections
<b>Panels</b> Individual Notices	On Click – expands to show full contents as below

## EXPANDED NOTICE



 | My Feed

View ▾

My Profile

Logout

Filter

ApplyClear

Date

From (Optional)  
yyyy/mm/dd

Until (Optional)  
yyyy/mm/dd

My Subscriptions

☐ Chapel Choir

☐ Investment Club

☐ Lost and Found

☐ Snell Society

Lost

Friday 21 Feb ▾

Lost

Friday 21 Feb ▾

B Block Service Representatives

Tuesday 25 Feb ▾

All B Block Boys

Tuesday 25 Feb ▾

Lost and Found

Sunday 11 Oct ▾

Clubs and Societies ▾

Snell Society

Friday 21 Feb ▾

Investment Club

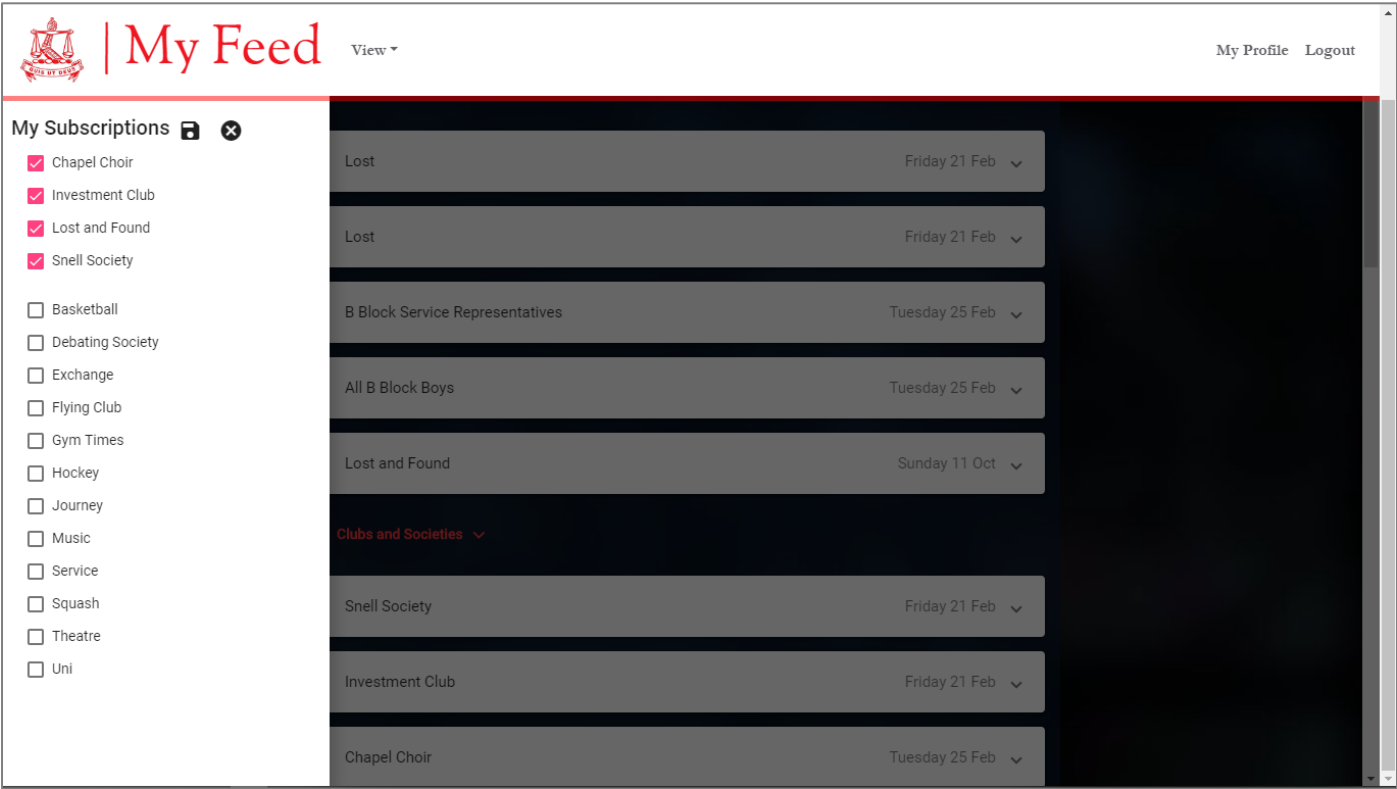
Friday 21 Feb ▾

Chapel Choir

Tuesday 25 Feb ▾


UI Element	Function
Button “Apply”	Displays the notices with the selected date filter applied
Button “Clear”	Removes filter from notices
Date Pickers “Date”	Allows user to specify a date range to filter the notices shown
Button “Pencil”	Invokes Subscription editor – see below
Check-boxes Subscription Names	Allows user to filter their notices by notification group

SUBSCRIPTION EDITOR



UI Element	Function
Button “Floppy Disk”	Allows user to save the changes to their subscriptions
Button “X Mark”	Allows user to save the changes to their subscriptions
Checkboxes Group names	Allows user to add or remove subscriptions

VIEW ALL NOTICES TABLE



All

Manage ▾ Publish ▾ View ▾

My Profile Logout

Date	Group	Title	Description
21 Feb	Uni	University Visits	Who: Crimson - They help to get sports scholarships at universities in the USA. When: **Today** @ 14h05. Where: LO Classrooms.
21 Feb	Lost and Found	Lost	An apple watch and an analogue watch were left in indoor centre during Wednesday's inter-house basketball. If they belong to you see Mr Sliedrecht in the Science office.
21 Feb	Chapel Choir	Chapel Choir	Choir workshop times for the weekend will be as follows: **Today** 15h00 - 17h30 and Sunday 10h30 - 13h00.
21 Feb	Lost and Found	Lost	school bag belonging to Callum James. Please speak to Mr Forward if found.
21 Feb	Gym Times	<div>Gym Times</div> <div>Keep</div> <div>Discard</div>	<div>05h30 - 06h15 14h45 - 17h30 20h20 - 21h00</div> <div>KeepDiscard</div>
21 Feb	Debating Society	Debating Society	meeting on Sunday directly after chapel in Mr Smith Classroom. We will be focusing on impromptu speech training in preparation for inter-house events. Anyone interested in learning a bit about this skill is welcome.
21 Feb	Snell Society	Snell Society	there is a Snell Society meeting this Sunday at 19h45 in the Centenary Centre. Nathan Bau and Sam Frost will be doing their presentations. Staff and boys are encouraged to attend.

Save Changes

UI Element	Location	Function
Cell Contents	Columns: [Title]; [Description]	On click (Each notice if Admin, only own notices if Staff) – show text area where user can edit the notice
Button “Keep”	Cell Editing	Confirm edits
Button “Discard”	Cell Editing	Revert edits
Button “Save Changes”	Bottom Right	Updates the `notices` Table with any changes

STAFF EDITING VARIATION

21 Feb

Gym Times

Gym Times

Request

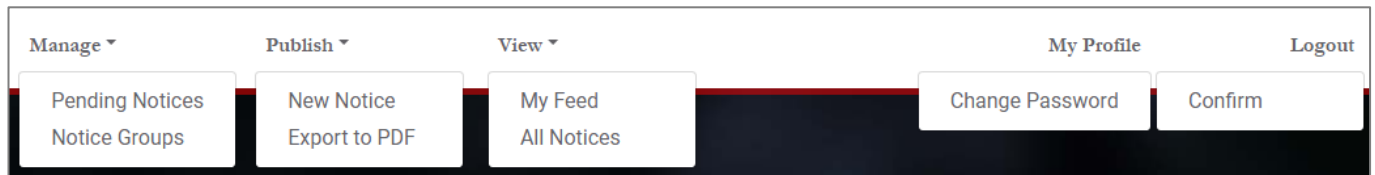
Discard

05h30 - 06h15  
14h45 - 17h30  
20h20 - 21h00

RequestDiscard


UI Element	Location	Function
Button “Request”	Cell Editing	Request Admin approval of Edits

## HEADER DROP DOWNS



UI Element	Function
<b>Hyperlink</b> “Pending Notices”	Directs user to the “Pending” webpage
<b>Hyperlink</b> “Notice Groups”	Directs user to the “Groups” webpage
<b>Hyperlink</b> “New Notice”	Directs user to the “New” webpage
<b>Hyperlink</b> “Export to PDF”	Directs user to the “Export” webpage
<b>Hyperlink</b> “My Feed”	Directs user to the “My Feed” webpage
<b>Hyperlink</b> “All Notices”	Directs user to the “View All” webpage
<b>Hyperlink</b> “Change Password”	Directs user to the “Change Password” webpage
<b>Hyperlink</b> “Confirm”	Logs user out and directs to the “Login” webpage

CHANGE PASSWORD FORM

 | 

# Change Password

[Manage](#) [Publish](#) [View](#)

[My Profile](#) [Logout](#)

Old Password

.....

New Password

.....

Confirm New Password

.....

Submit

UI Element	Function
<b>Text Field</b> “Old Password”	Allows user to enter their current password
<b>Text Field</b> “New Password”	Allows user to enter their new password
<b>Text Field</b> “Confirm Password”	Allows user to verify their new password
<b>Button</b> “Update”	If details valid – Updates the Users table and directs user to their previous page If details invalid – formats the dialog box as below

Old Password

.....

New Password

.....

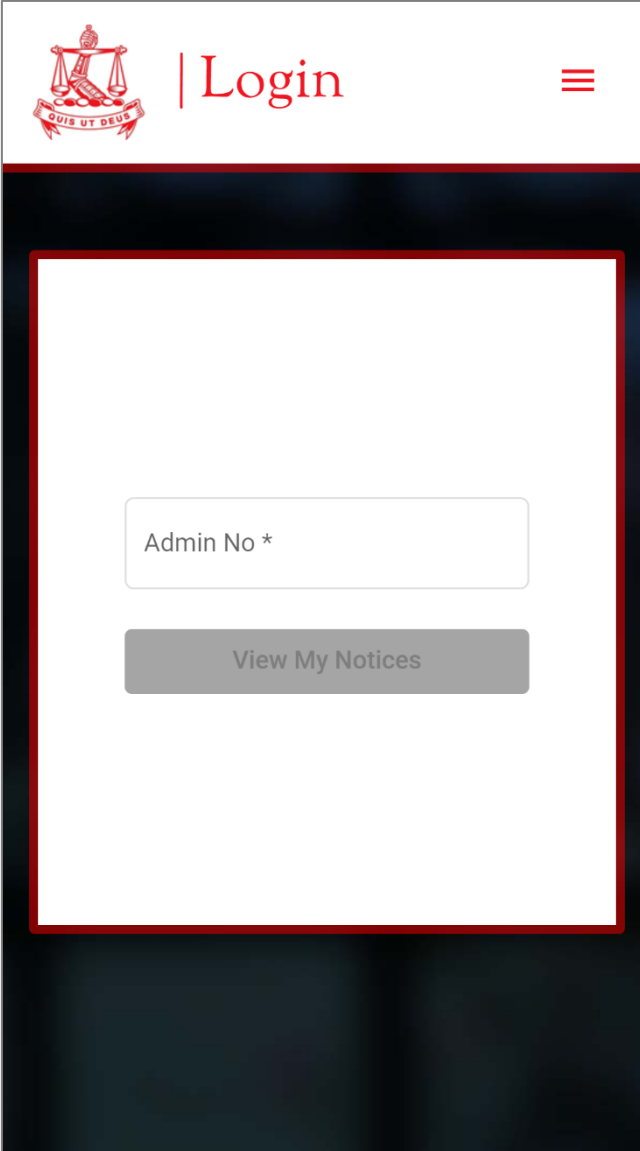
Confirm New Password

.....

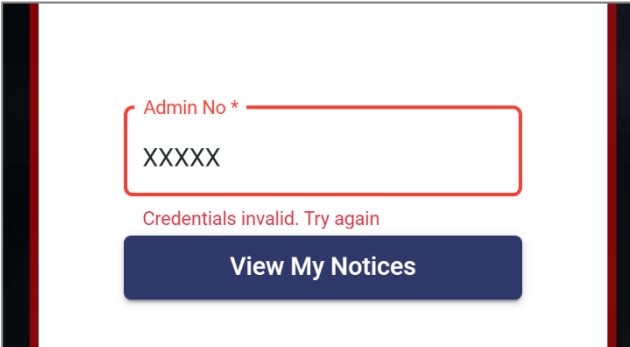
Invalid Credentials. Try again

Submit

## SIGN IN FORM

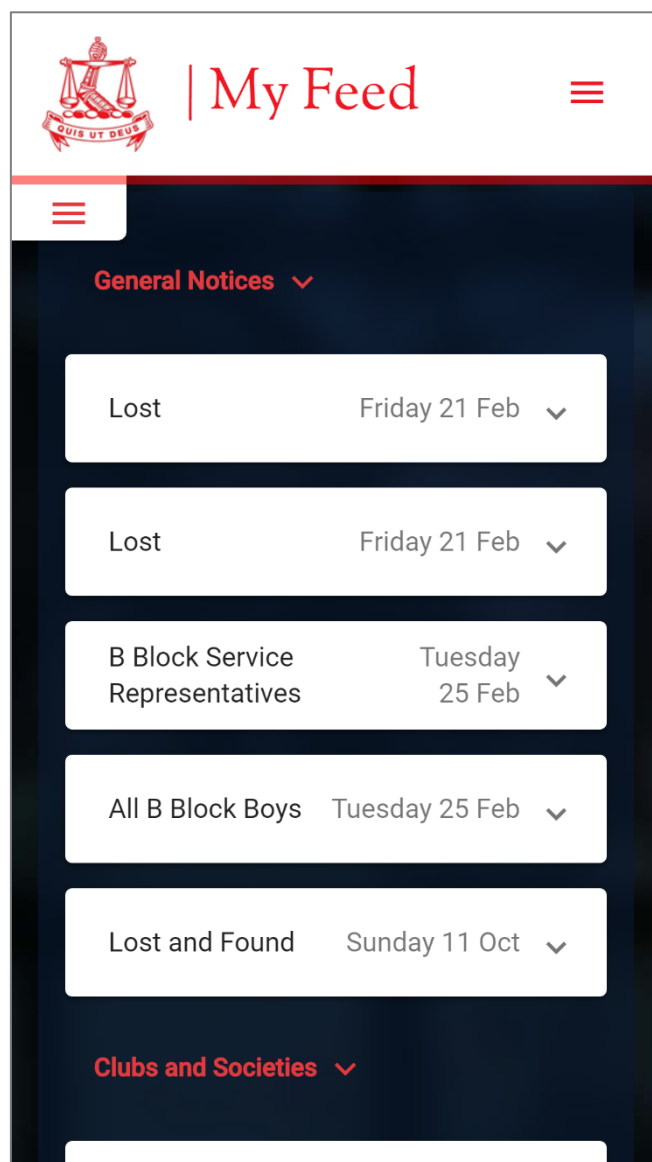
A mobile application login screen. At the top, there is a header bar with a red crest logo on the left, the text "| Login" in the center, and a red hamburger menu icon on the right. Below the header is a dark blue background. In the center, there is a white rectangular box with a red border. Inside this box, there is a text input field labeled "Admin No \*" and a grey button labeled "View My Notices".

UI Element	Function
<b>Text Field</b> "Admin No"	Allows user to enter their Student Administration Number
<b>Button</b> "View My Notices"	If details valid – directs user to the "My Feed" webpage If details invalid – formats page as below

A mobile application login screen showing an error state. The header bar is the same as the previous screenshot. The white box with the red border contains a text input field labeled "Admin No \*" with the text "XXXXX" inside. Below the input field, there is a red error message: "Credentials invalid. Try again". At the bottom of the box is a dark blue button labeled "View My Notices".

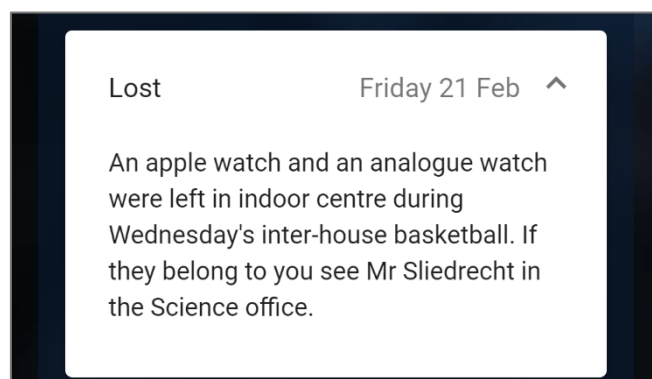




## PERSONALISED NOTICES FEED



UI Element	Function
<b>Hamburger Menu</b>	Reveals/Conceals the sidebar
<b>Buttons</b> Section Titles	Allows user to expand and collapse specific sections
<b>Cards</b> Individual Notices	On Tap – expands to show full contents as below

## EXPANDED NOTICE



 | My Feed 

Filter

Apply

Clear

Date

From (Optional)  
yyyy/mm/dd

Until (Optional)  
yyyy/mm/dd

My Subscriptions

☐ Chapel Choir

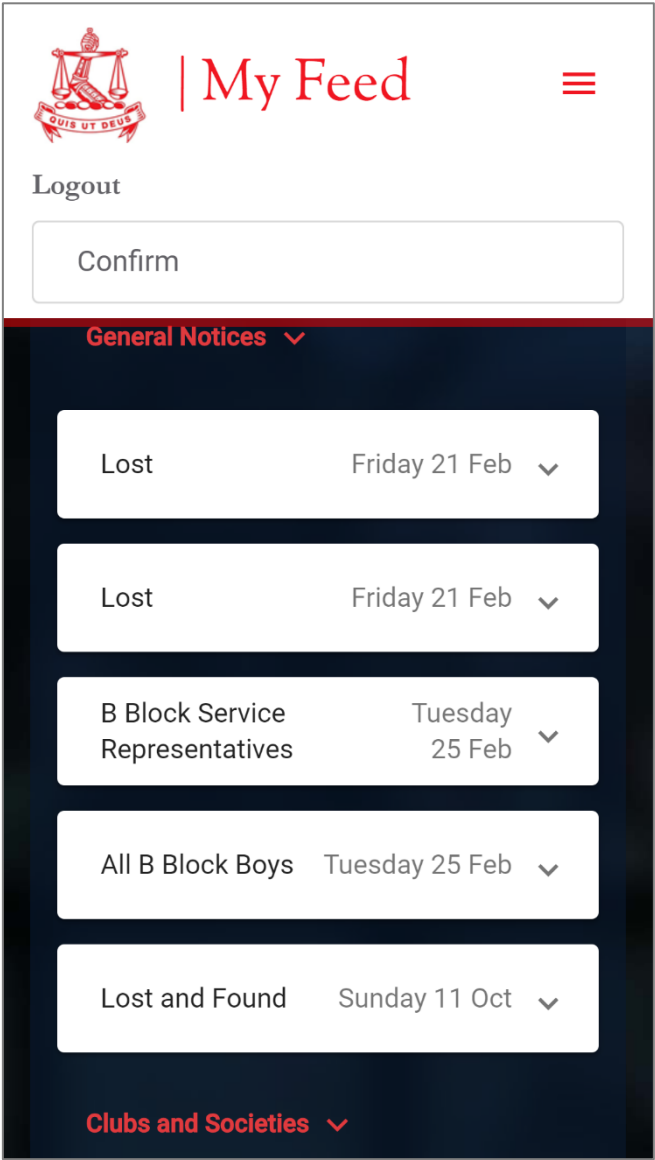
☐ Investment Club

☐ Lost and Found

☐ Snell Society

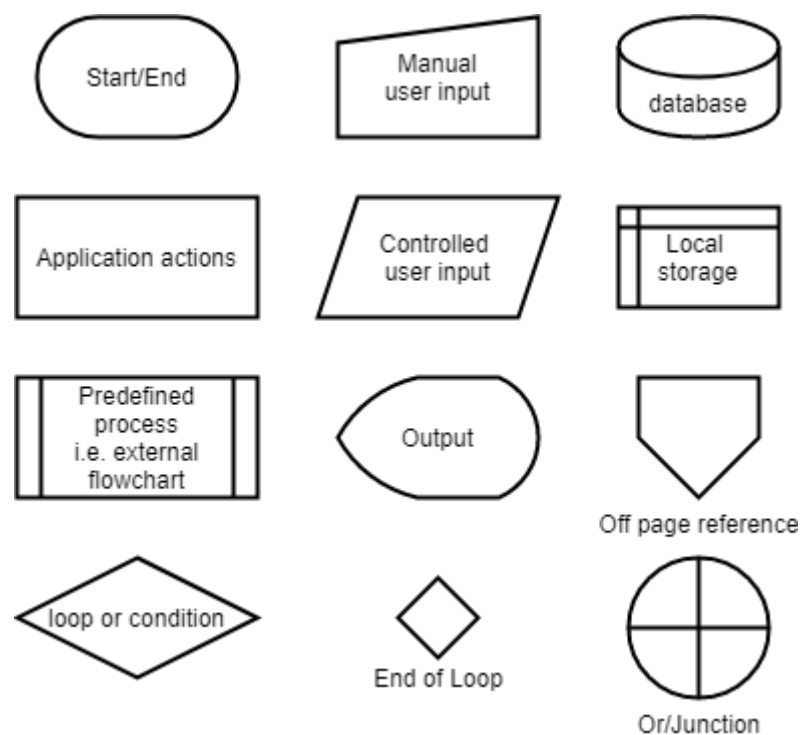
UI Element	Function
Button “Apply”	Displays the notices with the selected date filter applied
Button “Clear”	Removes filter from notices
Date Pickers “Date”	Allows user to specify a date range to filter the notices shown
Checkboxes Subscription Names	Allows user to filter their notices by notification group

HEADER MENU

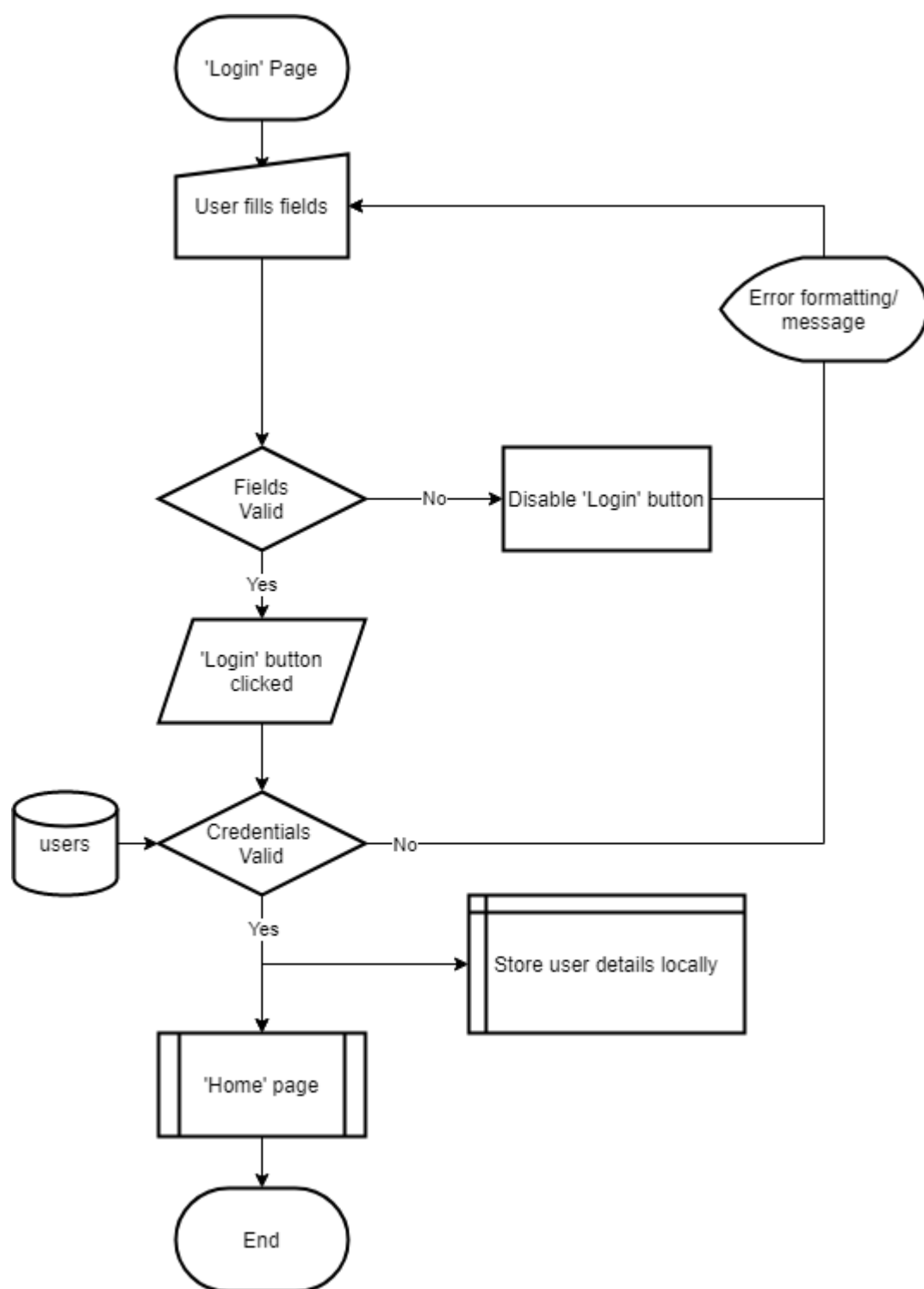


UI Element	Function
Hamburger Menu	Reveals/Conceals the collapsible menu
Button “Logout”	Opens Logout dropdown
Button “Confirm”	Logs user out and directs to the “Login” webpage

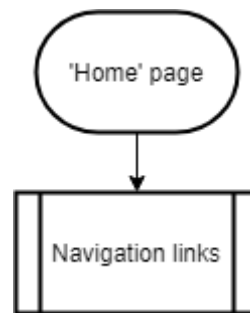
## FLOWCHART LEGEND



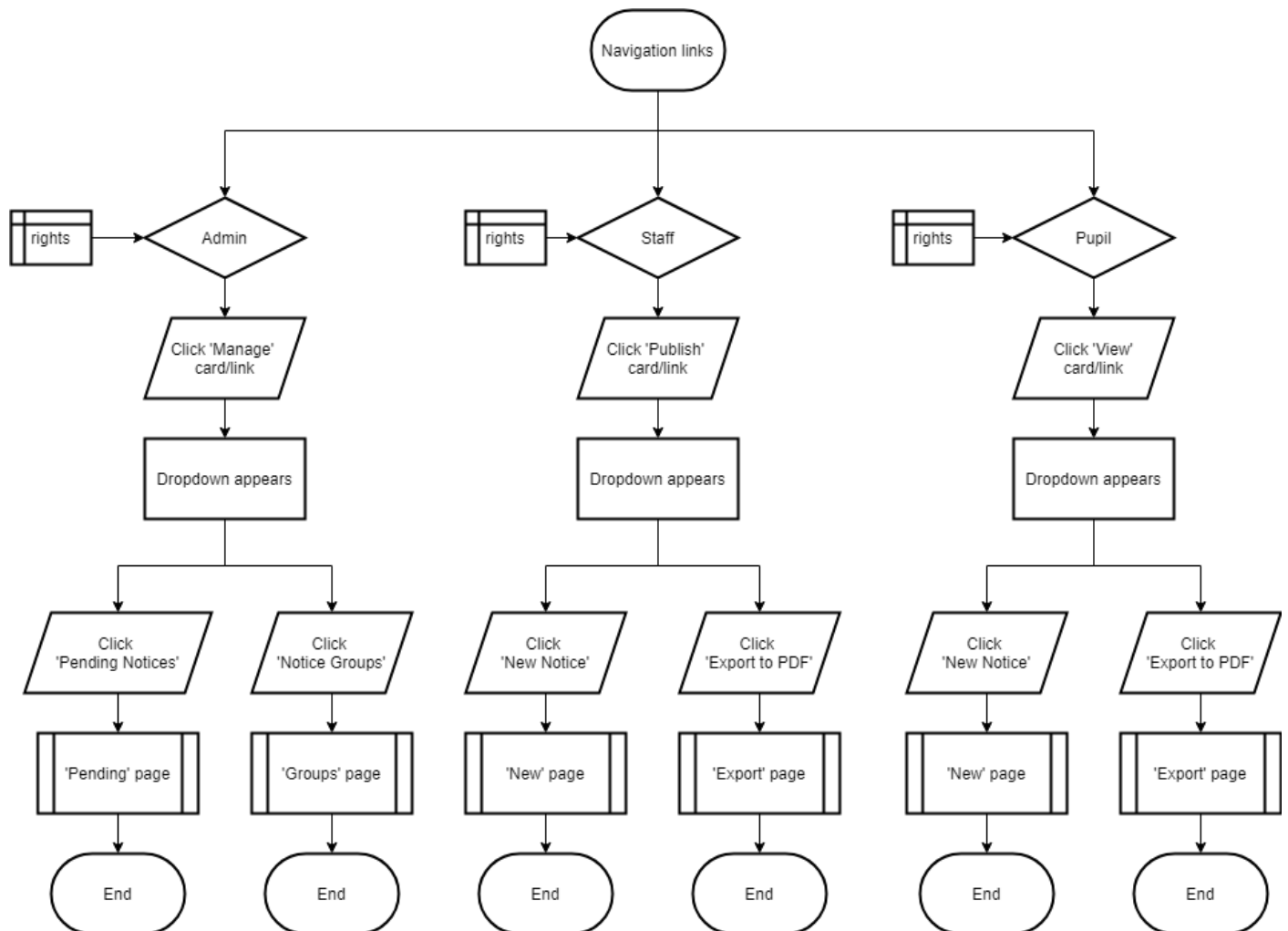
## LOGIN PAGE

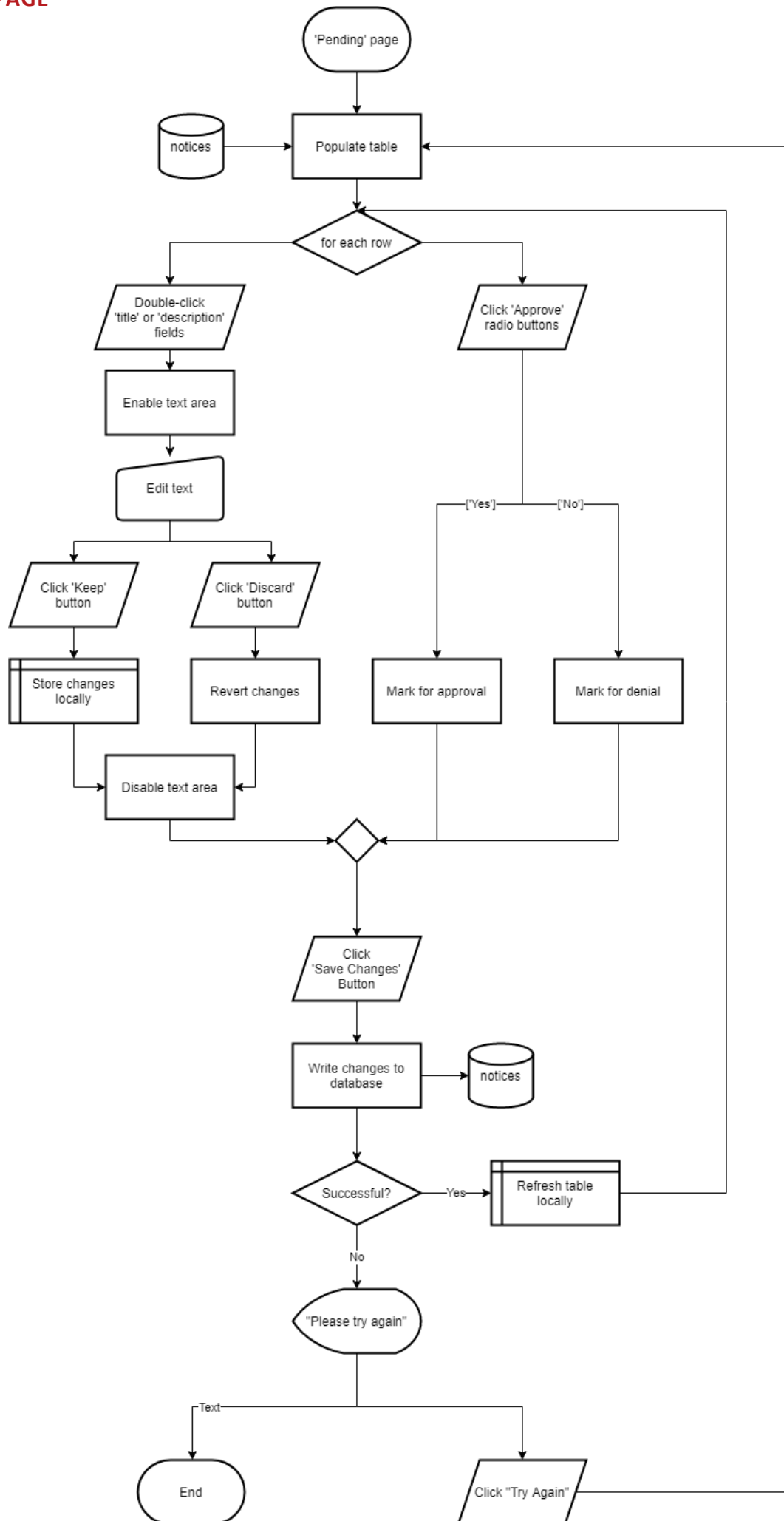


## HOME PAGE

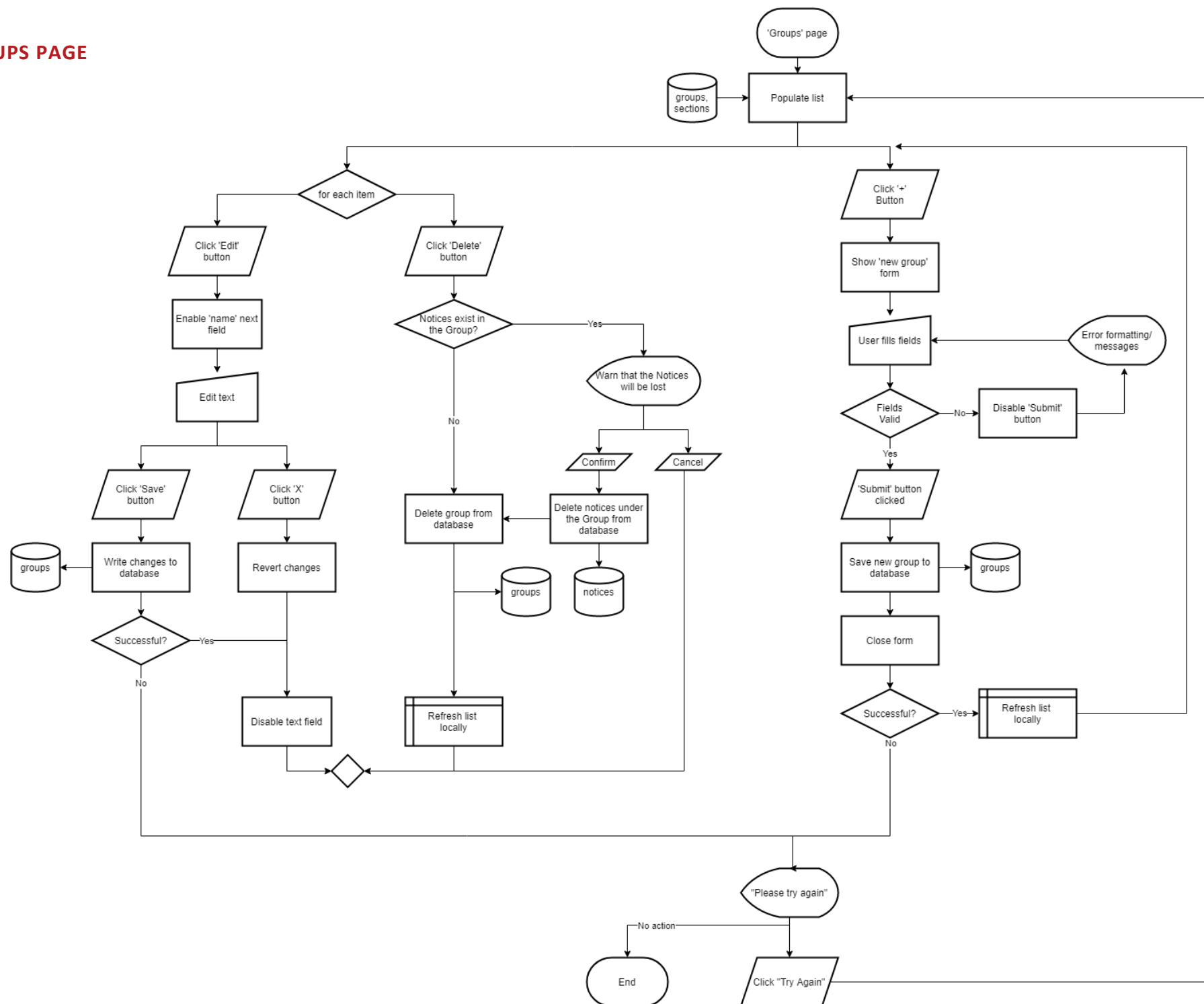


## NAVIGATION LINKS

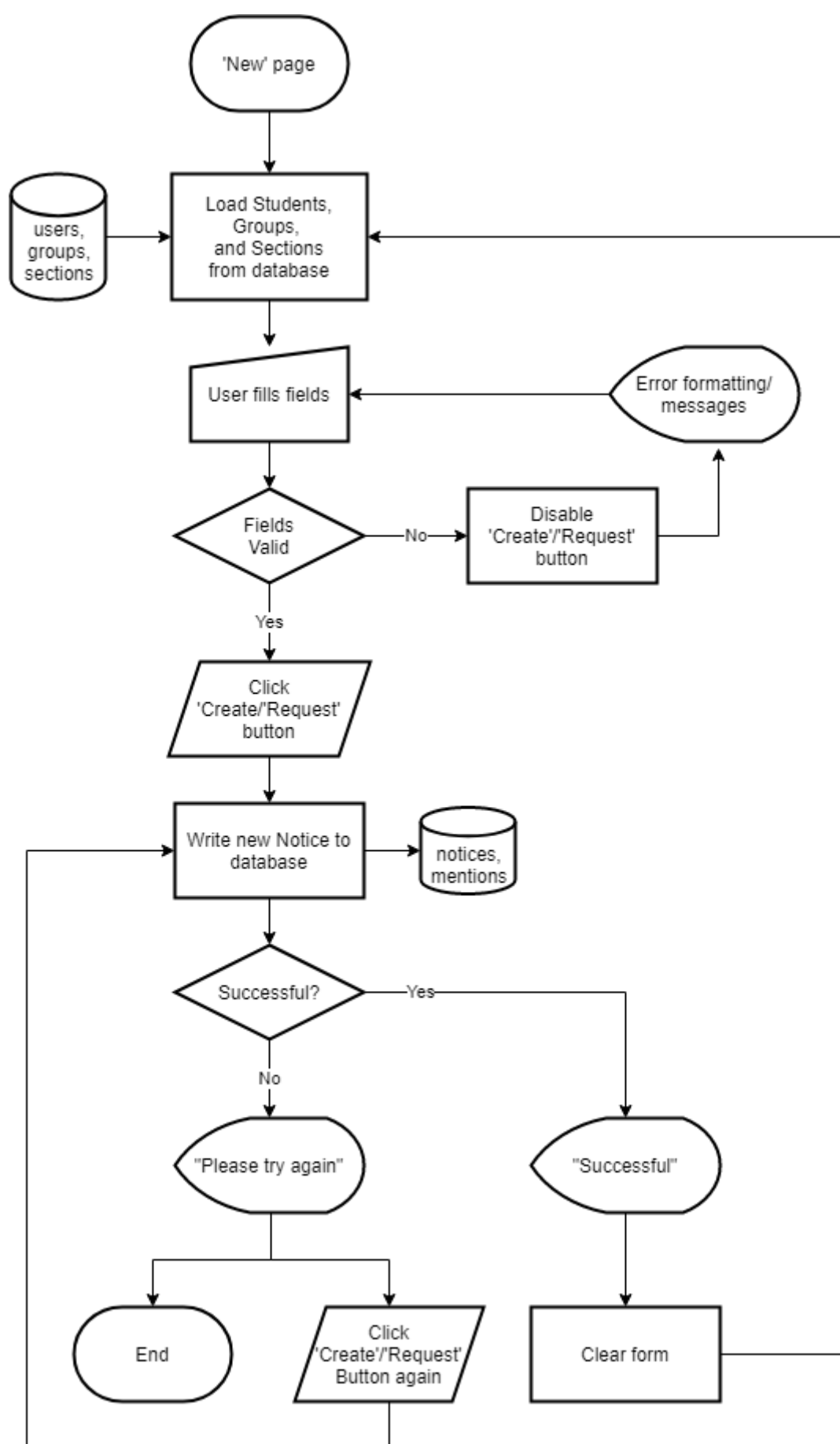


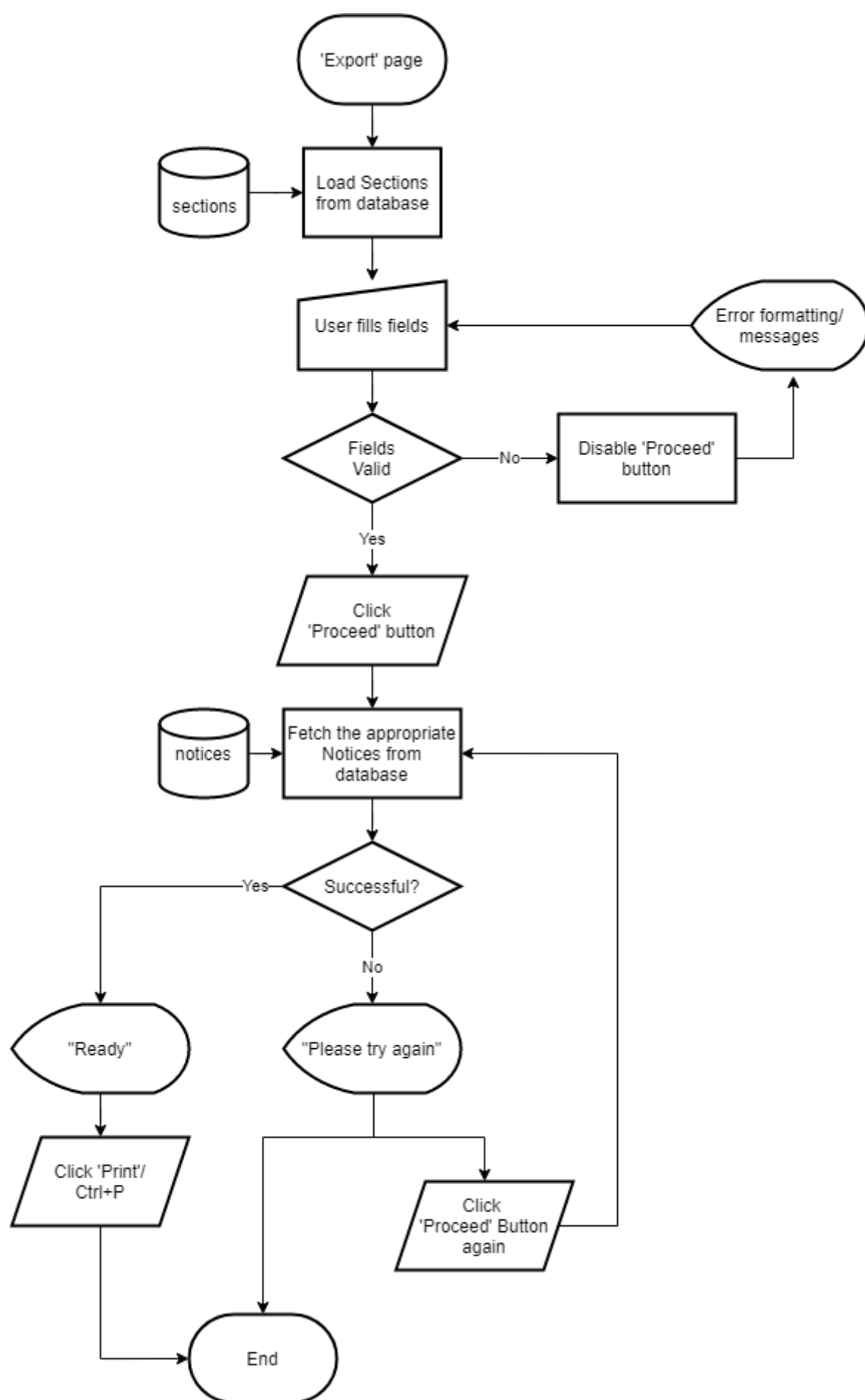


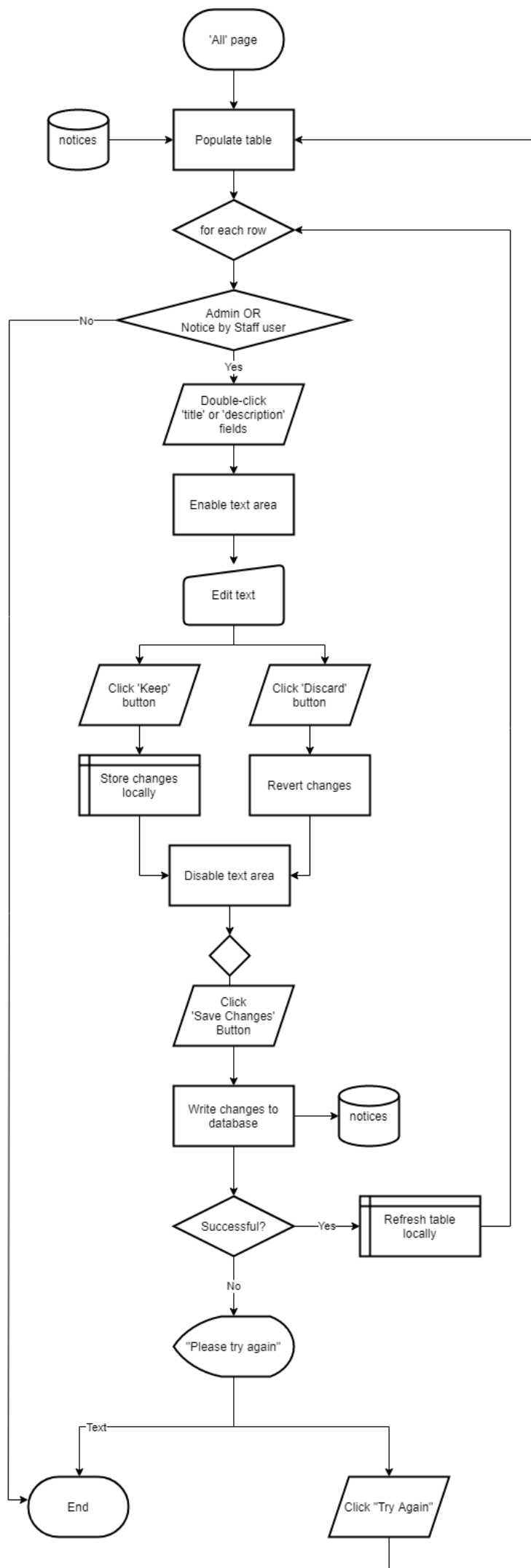
## GROUPS PAGE

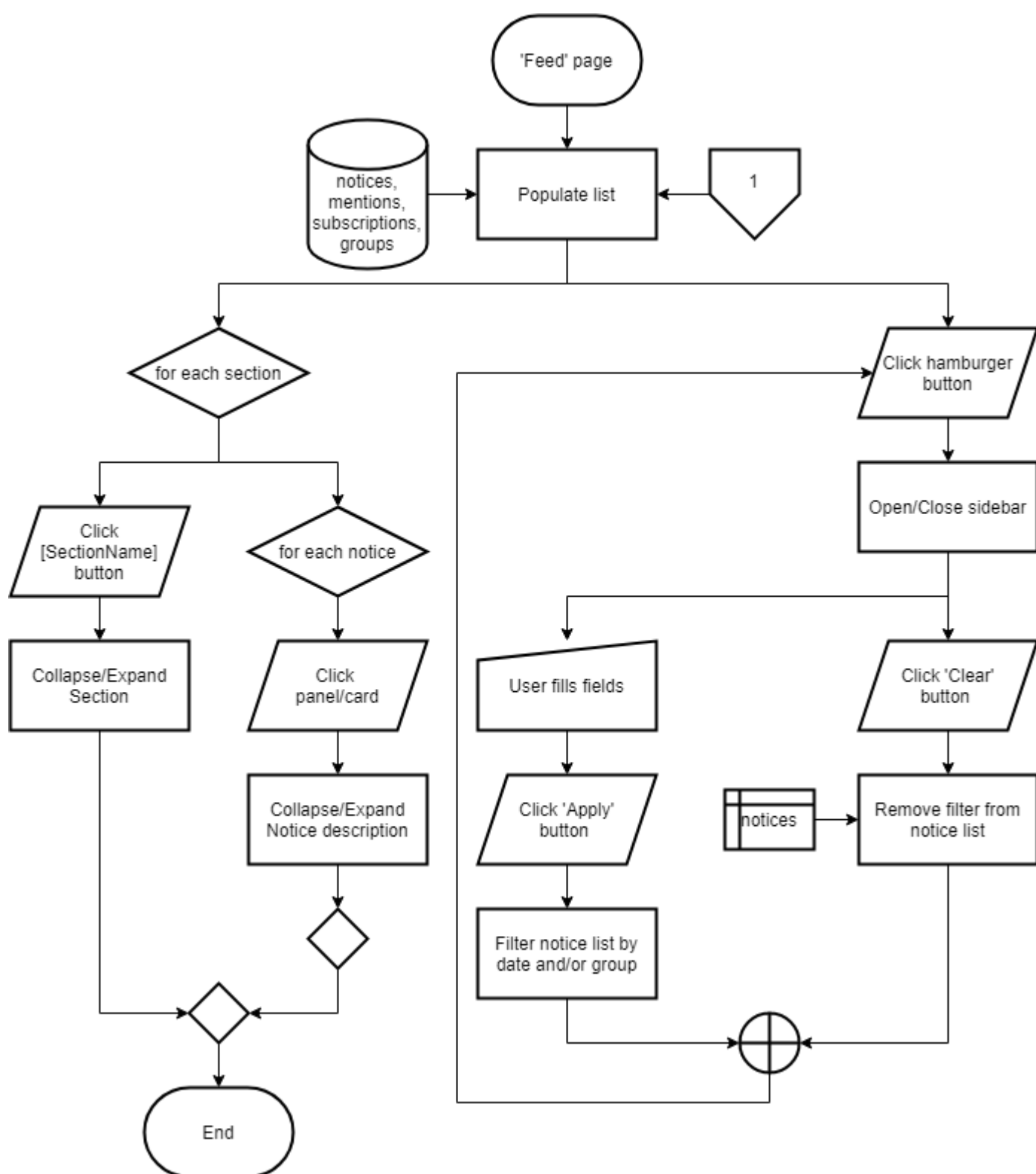


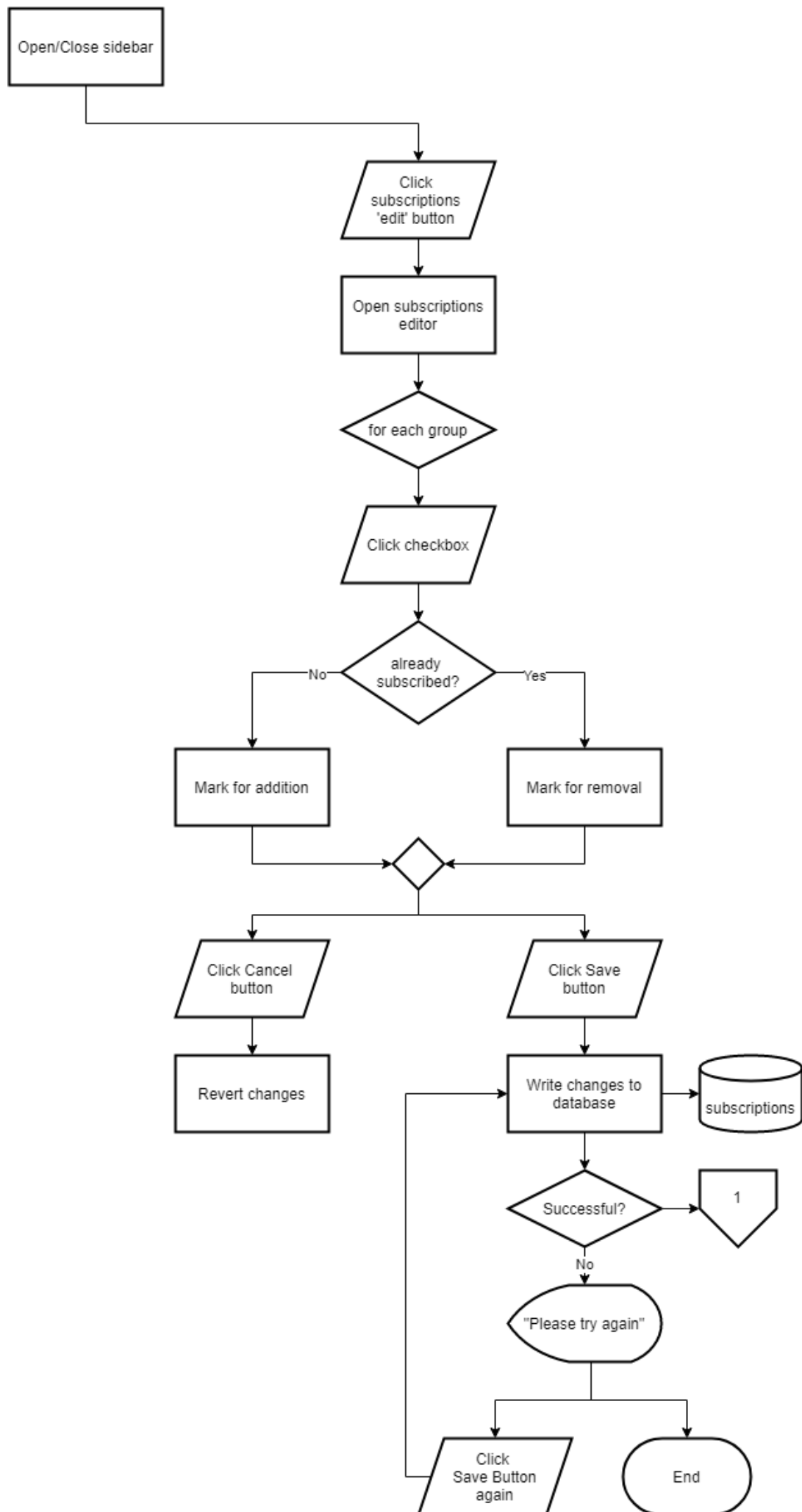




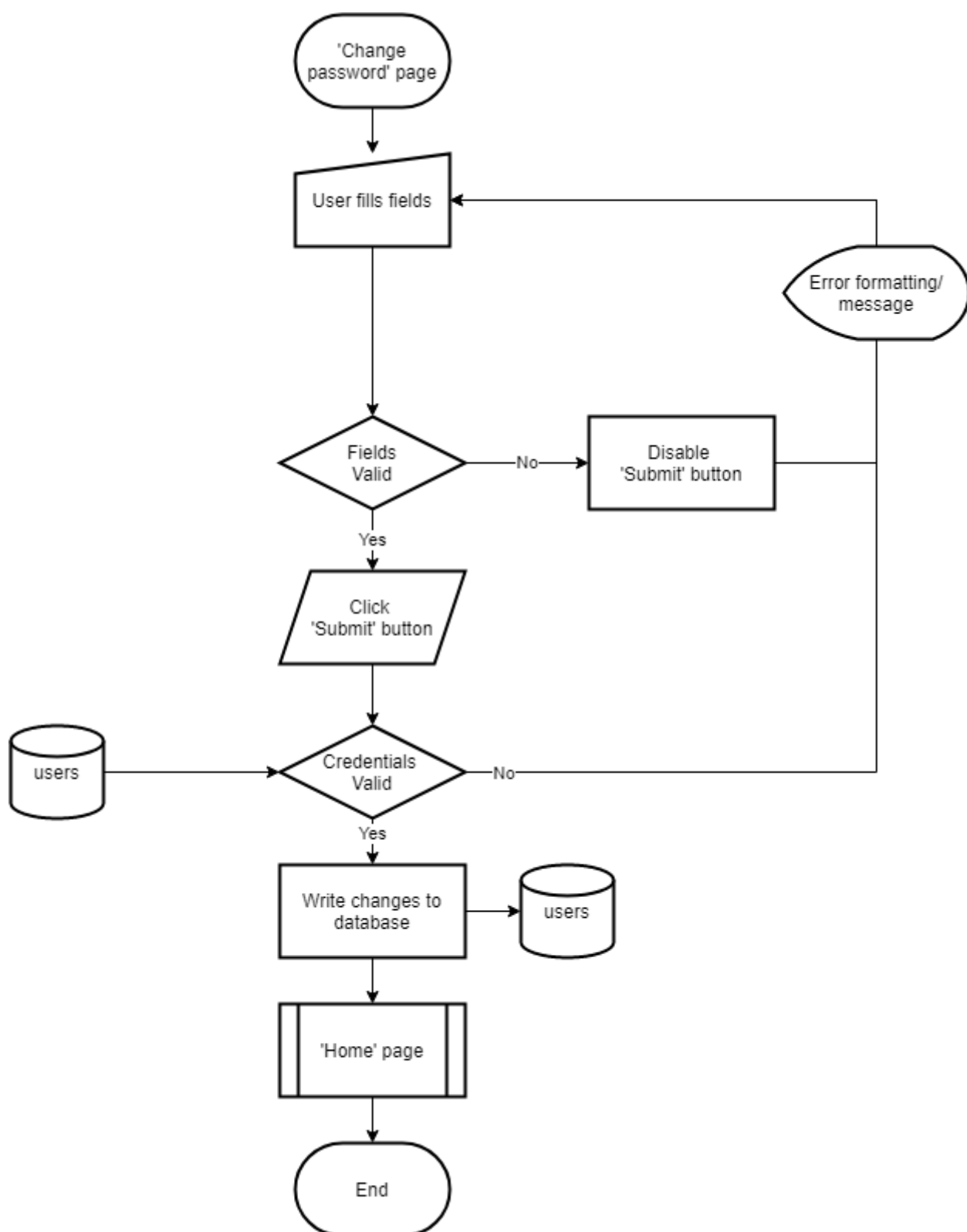




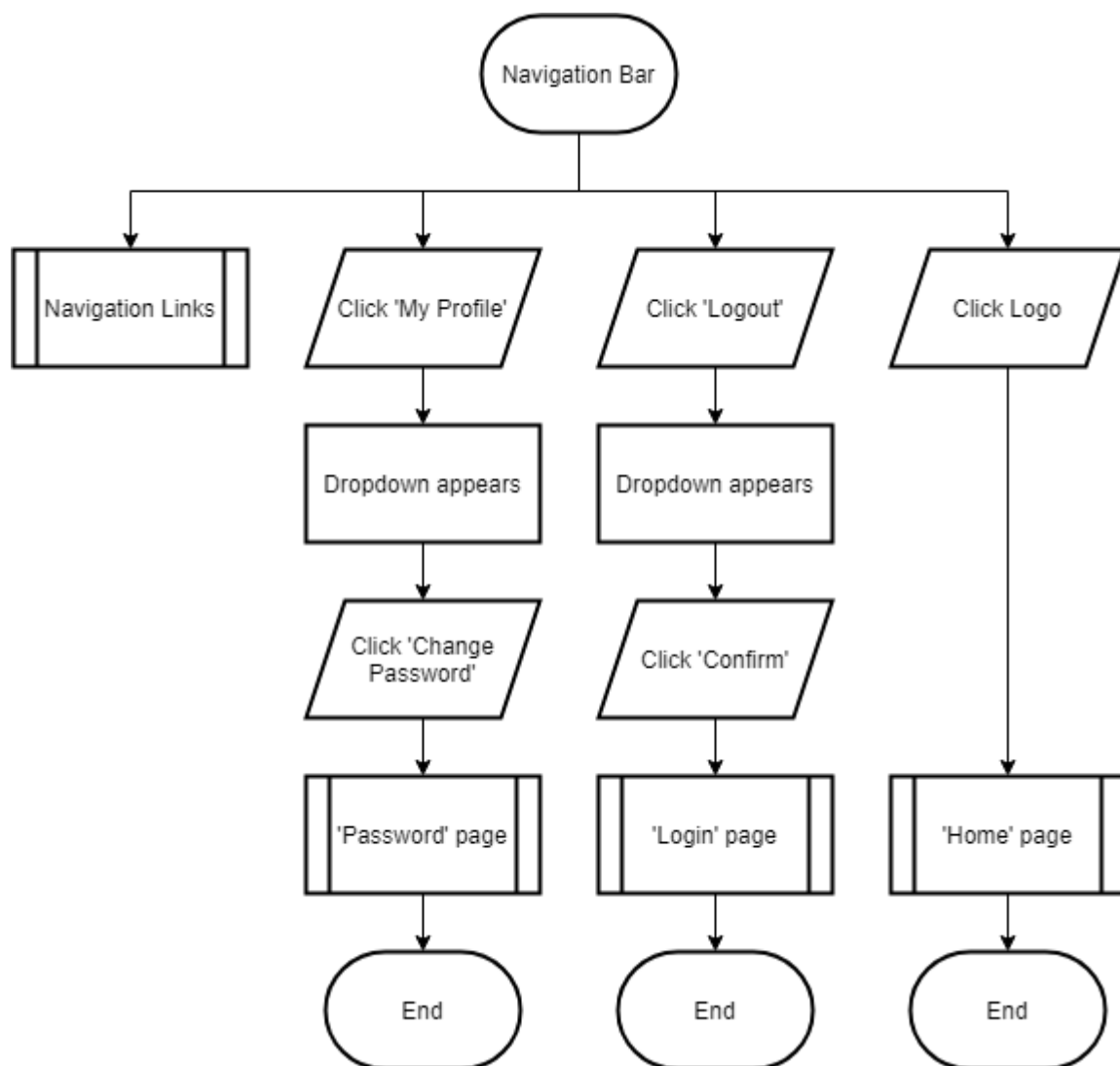




## PASSWORD PAGE



## NAVIGATION BAR

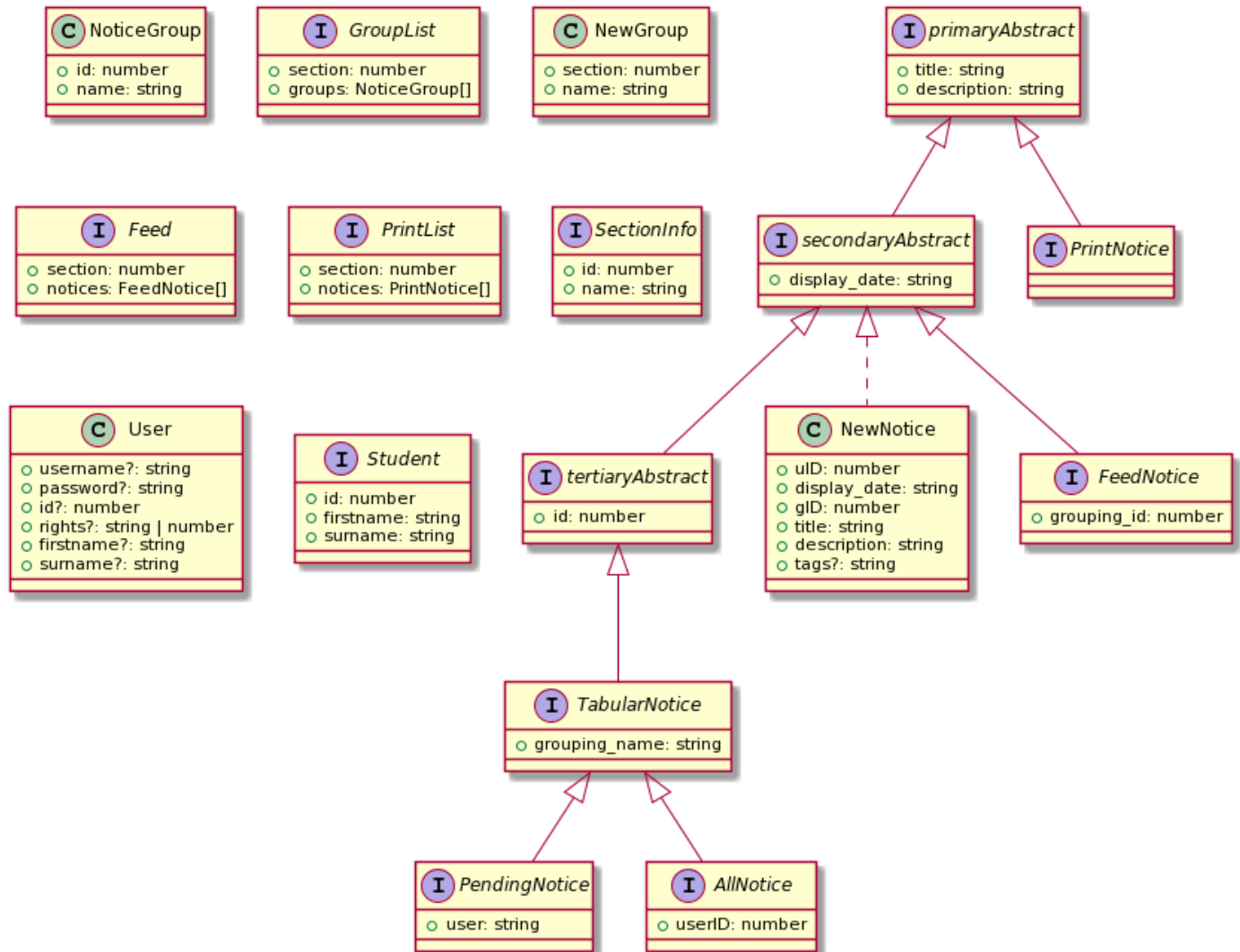


From here onwards, all documentation will presume familiarity with the fundamentals of MySQL, NodeJS, JavaScript/TypeScript, RxJS, and Angular.

(Please note: as per JS convention, “integers” are referred to as “numbers”)

## TYPESCRIPT INTERFACES

### OVERVIEW





## NOTICE TYPES

primaryAbstract	
Fields	
+ title : string	Stores the notice's title
+ description : string	Stores the notice's description

secondaryAbstract extends primaryAbstract	
Fields	
+ display_date : string	Stores the notice's display date

teritaryAbstract extends secondaryAbstract	
Fields	
+ id : number	Stores the notice's id

PrintNotice extends primaryAbstract	
-------------------------------------	--

NewNotice implements secondaryAbstract	
Fields	
+ uID : number	Stores the id of the user who generated the notice
+ display_date : string	Stores the notice's display date
+ gID : number	Stores the notice's grouping_id
+ title : string	Stores the notice's title
+ description : string	Stores the notice's description
+ tags? : string (optional)	Stores the comma-separated ids of students tagged in the notice
Methods	
+ constructor( uID: number, display_date: string, gID: number, title: string, description: string, tags?: string )	Instantiates an object of this class

FeedNotice extends secondaryAbstract	
Fields	
+ grouping_id : number	Stores the notice's grouping_id

TabularNotice extends tertiaryAbstract	
Fields	
+ grouping_name : string	Stores the name of the group under which the notice falls

AllNotice extends TabularNotice	
Fields	
+ userID : number	Stores the id of the user who generated the notice

PendingNotice extends TabularNotice	
Fields	
+ user : number	Stores the name of the user who generated the notice

PrintList	
Fields	
+ section : number	Stores the id of a particular section
+ notices : PrintNotice[]	Stores the notices falling under this section

Feed	
Fields	
+ section : number	Stores the id of a particular section
+ notices : FeedNotice[]	Stores the notices falling under this section

## GROUP TYPES

NoticeGroup	
Fields	
+ id : number	Stores the group's id
+ name : string	Stores the group's name

GroupList	
Fields	
+ section : number	Stores the id of a particular section
+ groups : NoticeGroup[]	Stores the groups falling under this section

NewGroup	
Fields	
+ section : number	Stores the id of the section under which the group falls
+ name : string	Stores the group's name
Methods	
+ constructor(section: number, name: string)	Instantiates an object of this class

## SECTION

SectionInfo	
Fields	
+ id : number	Stores the section's id
+ name : string	Stores the section's name

## USER TYPES

Student	
Fields	
+ id : number	Stores the student's id
+ firstname : string	Stores student's first name
+ surname : string	Stores student's surname

User	
Fields	
+ username? : string (optional)	Stores the user's username
+ password? : string (optional)	Stores the user's password
+ id? : number (optional)	Stores the user's id
+ rights? : number   string (optional)	Stores the user's authorisation rights
+ firstname? : string (optional)	Stores the user's first name
+ surname? : string (optional)	Stores the user's surname
Methods	
+ constructor( username?: string, password?: string, id?: number, rights?: number   string, firstname?: string, surname?: string )	Instantiates an object of this class

## BACKEND – NODE (EXPRESS) REST SERVER

### MYSQL DATABASE

db	
Methods	
+ getPending() : Promise<PendingNotice[]>	Returns an Array containing all pending notices from the database
+ setNoticeStatus(id: number, status: number) : Promise<any>	Updates the approval_status enum of a notice
+ getGroups() : Promise<GroupList[]>	Returns an Array containing all groups from the database
+ setGroupName(id: number, name: string) : Promise<any>	Updates the name of a group

+ deleteGroup(id: number, recursive: boolean) : Promise<any>	Deletes a group from the database if possible (recursive == false)  Deletes all notices within a group and the group itself (recursive == true)
+ addGroup(section: number, name: string) : Promise<any>	Inserts a new group into the `groupings` table according to the information passed
+ getSections() : Promise<SectionInfo[]>	Returns an Array containing all sections from the database
+ newNotice( userID: number, displayDate: string, groupingID: number, title: string , description: string, rights: number ) : Promise<any>	Inserts a new notice into the `notices` table according to the information passed  If Staff – sets approval_status to “Pending”  If Admin – sets approval_status to “Approved”
+ getLastNoticeID() : Promise<number>	Returns number storing the LAST_INSERT_ID() from the database
+ tagStudents(noticeID: number, idsToMention: string) : Promise<any>	Iterates through the idsToMention string to insert relevant tuples into the `mentions` table
+ getStudents() : Promise<Student[]>	Returns an Array containing all students from the database
+ exportNotices(date: string, sections: string) : Promise<PrintList[]>	Returns an Array containing approved notices which satisfy the criteria passed
+ getFeed(userID: number) : Promise<Feed[]>	Returns an Array containing approved notices which either fall under the user’s subscription groups, mention the user, or have not been placed in any specific predefined group (as will likely be the case for Restrictions, To See Staff, and Sanatorium notices) from the database
+ getSubGroups(userID: number) : Promise<NoticeGroup[]>	Returns an Array containing groups from the database to which the user has and has not subscribed (in that order), with the first element representing the number of subscriptions
+ addSubs(userID: number, subsToAdd: string) : Promise<any>	Iterates through the subsToAdd string to insert relevant tuples into the `subscriptions` table
+ deleteSubs(userID: number, subsToDel: string) : Promise<any>	Iterates through the subsToDel string to delete relevant tuples from the `subscriptions` table
+ getAll() : Promise<AllNotice[]>	Returns an Array containing all approved notices from the database
+ editNotice(id: number, title: string, description: string, rights: number) : Promise<any>	Updates the title and/or description of a notice  If Staff – sets approval_status to “Pending”

+ login(username: string, password: string) : Promise<any>	If the credentials passed are valid, returns an object containing the user's details from the database and a token thereof
+ mobileLogin(adminNo: number) : Promise<any>	If the admin number passed is valid, returns an object containing the student's details from the database and a token thereof
+ changePassword(username: string, oldPassword: string, newPassword: string) : Promise<any>	If the credentials passed are valid, updates the password field of a user's tuple in the `users` table

## EXPRESS ROUTER

router	
Methods	
- verifyAdmin(req: Request, res: Response, next: NextFunction) : any	Verifies the token of an admin accessing a route and allows HTTP request to proceed if token valid, otherwise sends an error 401 (Unauthorised)
- verifyStaff(req: Request, res: Response, next: NextFunction) : any	Likewise, for staff users
- verifyPupil(req: Request, res: Response, next: NextFunction) : any	Likewise, for pupil users on desktop
- verifyMobile(req: Request, res: Response, next: NextFunction) : any	Likewise, for pupil users on mobile
+ get("/pending", verifyAdmin, callback: RequestHandler<>) : Response<any>	Returns an HTTP 200 (OK) response with the results of calling db.getPending(), otherwise an error 500 (Internal Server Error)
+ put("/pending", verifyAdmin, callback: RequestHandler<>) : Response<any>	Likewise, after calling <b>db.setNoticeStatus(...)</b>
+ get("/groups", verifyStaff, callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling db.getGroups()
+ put("/groups", verifyAdmin, callback: RequestHandler<>) : Response<any>	Likewise, after calling <b>db.setGroupName(...)</b>
+ delete("/groups", verifyAdmin, callback: RequestHandler<>) : Response<any>	Likewise after calling <b>db.deleteGroup(...)</b>
+ post("/groups", verifyAdmin, callback: RequestHandler<>) : Response<any>	Likewise, after calling <b>db.newGroup(...)</b>
+ put("/sections", verifyMobile, callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling <b>db.getSections()</b>
+ post("/new", verifyStaff, callback: RequestHandler<>) : Response<any>	Likewise, after calling <b>db.newNotice(...)</b> – and, if there are students to tag, <b>db.getLastNoticeID()</b> and <b>db.tagStudents(...)</b>

+ get("/students", verifyStaff, callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling <b>db.getStudents()</b>
+ get("/export/:date/:sections", verifyStaff, callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling <b>db.exportNotices(...)</b>
+ get("/feed/:id", verifyPupil, callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling <b>db.getFeed(...)</b>
+ get("/mobile/feed", verifyMobile, callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling <b>db.getFeed(...)</b>
+ get("subs/:id", verifyMobile, callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling <b>db.getSubGroups(...)</b>
+ post("/subs/:id/:add", verifyPupil, callback: RequestHandler<>) : Response<any>	Likewise, after calling <b>db.addSubs(...)</b>
+ delete("/subs/:id/:remove", verifyPupil, callback: RequestHandler<>) : Response<any>	Likewise, after calling <b>db.deleteSubs(...)</b>
+ get("/all", verifyPupil, callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling <b>db.getAll()</b>
+ put("/edit", verifyStaff, callback: RequestHandler<>) : Response<any>	Likewise, after calling <b>db.editNotice(...)</b>
+ post("/login", callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling <b>db.login(...)</b> , otherwise an error 401
+ post("/mobile/login", callback: RequestHandler<>) : Response<any>	Likewise, with the results of calling <b>db.mobileLogin(...)</b>
+ post("/password", verifyPupil, callback: RequestHandler<>) : Response<any>	Likewise, after calling <b>db.changePassword(...)</b>

## NGMODULES

*\*(From Angular Documentation)*

AppModule	
Fields	
- <u>declarations</u> : (any[]   Type<any>)[ ]	"The set of components, directives, and pipes (declarables) that belong to this module."*
- <u>imports</u> : (any[]   Type<any>   ModuleWithProviders<{}>)[ ]	"The set of NgModules whose exported declarables are available to templates in this module."*
- <u>providers</u> : Provider[ ]	"The set of injectable objects that are available in the injector of this module."*
+ <u>bootstrap</u> : (any[]   Type<any>)[ ]	"The set of components that are bootstrapped when this module is bootstrapped."*

AppRoutingModule	
Fields	
- <u>imports</u> : (any[]   Type<any>   ModuleWithProviders<{}>)[ ]	"The set of NgModules whose exported declarables are available to templates in this module."*
+ <u>exports</u> : (any[]   Type<any>)[ ]	"The set of components, directives, and pipes declared in this NgModule that can be used in the template of any component that is part of an NgModule that imports this NgModule."*
- <u>routes</u> : Route[ ]	"Represents a route configuration for the Router service."*

MaterialModule	
Fields	
- <u>imports</u> : (any[]   Type<any>   ModuleWithProviders<{}>)[ ]	"The set of NgModules whose exported declarables are available to templates in this module."*
+ <u>exports</u> : (any[]   Type<any>)[ ]	"The set of components, directives, and pipes declared in this NgModule that can be used in the template of any component that is part of an NgModule that imports this NgModule."*
- <u>MaterialComponents</u> : any[ ]	The set of Angular Material components that can be used in this NgModule



## SERVICES

NoticeService	
Fields	
- <code>_dbUrl : string</code>	Stores the base URL of the express REST API server
- <code>_sections : SectionInfo[]</code>	Stores the names and ids of sections from the database after initially retrieved by the user. This is to limit the number of calls to the server
- <code>_students : Student[]</code>	Stores the names and ids of students from the database after initially retrieved by the user. This is to limit the number of calls to the server
- <code>http : HttpClient</code>	Injects the HttpClientService "to perform HTTP requests"*
- <code>snackBar : MatSnackBar</code>	Injects the MatSnackBarService "to dispatch Material Design snack bar messages"*
Methods	
+ <code>constructor(http: HttpClient, snackBar: MatSnackBar)</code>	Instantiates an object of this class and injects the passed dependencies
+ <code>getPending() : Observable&lt;PendingNotice[]&gt;</code>	Returns an Observable of the results of a GET request to <code>"/pending"</code> , otherwise an empty Observable to prevent crashing
+ <code>updatePendingStatus(id: number, status: number) : Observable&lt;any&gt;</code>	Likewise, of a PUT request to <code>"/pending"</code>
+ <code>getGroups() : Observable&lt;GroupList[]&gt;</code>	Likewise, of a GET request to <code>"/groups"</code>
+ <code>setGroupName(id: number, name: string) : Observable&lt;any&gt;</code>	Likewise, of a PUT request to <code>"/groups"</code>
+ <code>deleteGroup(id: number, recursive: boolean) : Observable&lt;any&gt;</code>	Likewise, of a DELETE request to <code>"/groups/:id/:recursive"</code>
+ <code>addGroup(group: NewGroup) : Observable&lt;any&gt;</code>	Likewise, of a POST request to <code>"/groups"</code>
+ <code>getSections() : Observable&lt;SectionInfo[]&gt;</code>	Likewise, of a GET request to <code>"/sections"</code>
+ <code>newNotice(n: NewNotice) : Observable&lt;any&gt;</code>	Likewise, of a POST request to <code>"/new"</code>
+ <code>exportToPDF(date: any, sections: any) : Observable&lt;PrintList[]&gt;</code>	Likewise, of a GET request to <code>"/export/:date/:sections"</code>
+ <code>getMyNotices(id: number) : Observable&lt;Feed[]&gt;</code>	Likewise, of a GET request to <code>"/feed/:id"</code>
+ <code>getMyNoticesMobile() : Observable&lt;Feed[]&gt;</code>	Likewise, of a GET request to <code>"/mobile/feed"</code>
+ <code>getSubs(id: number) : Observable&lt;NoticeGroup[]&gt;</code>	Likewise, of a GET request to <code>"/subs/:id"</code>
+ <code>addSubs(id: number, add: string) : Observable&lt;any&gt;</code>	Likewise, of a POST request to <code>"/subs/:id/:add"</code>
+ <code>removeSubs(id: number, remove: string) : Observable&lt;any&gt;</code>	Likewise, of a GET request to <code>"/subs/:id/:remove"</code>
+ <code>getAll(): Observable&lt;AllNotice[]&gt;</code>	Likewise, of a GET request to <code>"/all"</code>

+ editNotice(id: number, title: string, description: string) : Observable<any>	Likewise, of a PUT request to "/edit"
- handleError<T>( operation: string = 'operation', result?: T ) : Observable<T>	Returns an empty observable of the specified type upon an error of an HTTP request. This is to prevent the application from crashing after such an error

AuthService	
Fields	
- _loginUrl : string	Stores the login url of the the express REST API server
- _mobileLoginUrl : string	Stores the mobile login url of the the express REST API server
- _passwordUrl : string	Stores the change password url of the the express REST API server
- _user : User	Stores an object with the user's details
- _loginObserver : Observer	Object with callback function common to both login methods
- http : HttpClient	Injects the HttpClientService "to perform HTTP requests"
- router : Router	Injects the RouterService for "navigation among views"
Methods	
+ constructor(http: HttpClient, router: Router)	Instantiates an object of this class and injects the passed dependencies
+ login(user: User) : Observable<any>	Returns an Observable of the results of a POST request to "/login", and navigates to "Home" if login successful
+ mobileLogin(adminNo: number) : Observable<any>	Likewise, of a POST request to "/mobile/login"
+ changePassword(oldPass: any, newPass: any) : Observable<any>	Likewise, of a POST request to "/password"
+ fetchDetails()	Attempts to assign an object of a user's details to the _user property, otherwise logs user out
+ logout()	Resets locally stored user details and navigates to "Login"
+ get isLoggedIn() : boolean	Returns a boolean of whether a visitor is logged in
+ get hasDetails() : boolean	Returns a boolean of whether the _user property has been initialised
+ get token() : string	Returns a string of the current user's jsonwebtoken (jwt)

+ get rights() : string   number	Returns the current user's authorisation rights
+ get userID() : number	Returns the current user's id

DeviceService	
Methods	
+ get web() : boolean	Returns a boolean of whether the visitor is a web user, based on the browser's navigator.userAgent
+ get mobile() : boolean	Returns a boolean of whether the visitor is a mobile user, based on the browser's navigator.userAgent

BaseResolverService	
Fields	
# ns : NoticeService	Injects the NoticeService to query the database indirectly for application information
# auth : AuthService	Injects the AuthService to perform authorisation-related tasks
Methods	
+ constructor(ns: NoticeService, auth: AuthService)	Instantiates an object of this class and injects the passed dependencies

[AllTable   Feed   Groups   Pending   Sections   Students   Subs]ResolverService extends BaseResolverService implements Resolve	
Methods	
+ resolve( route: ActivatedRouteSnapshot, state: RouterStateSnapshot)	Calls a relevant NoticeService method and "waits for the data to be resolved before the route is finally activated"*

TokenInterceptorService implements HttpInterceptor	
Fields	
- injector : Injector	"Concrete injectors implement this interface."*
Methods	
+ constructor(injector: Injector)	Instantiates an object of this class and injects the passed dependencies
+ intercept(req: any, res: any) : any	"Identifies and handles a given HTTP request,"* appending the current user's token to it

## GUARDS

<b>BaseGuard</b> implements CanActivate	
Fields	
# requiredRights : number	Stores the minimum authorisation rights for the guard
# auth : AuthService	Injects the AuthService to perform authorisation-related tasks
# router : Router	Injects the RouterService for “navigation among views”*
# snackBar : MatSnackBar	Injects the MatSnackBarService “to dispatch Material Design snack bar messages”*
Methods	
+ constructor(auth: AuthService, router: Router, snackBar: MatSnackBar)	Instantiates an object of this class and injects the passed dependencies
+ canActivate() : boolean	Returns a boolean of whether a user has permission to activate a requested route. If true, navigation continues. If false, navigation is cancelled.

<b>AdminGuard</b> extends BaseGuard implements CanActivate	
Fields	
+ requiredRights : number = 3	Stores the minimum authorisation rights for the guard

<b>StaffGuard</b> extends BaseGuard implements CanActivate	
Fields	
+ requiredRights : number = 2	Stores the minimum authorisation rights for the guard

<b>PupilGuard</b> extends BaseGuard implements CanActivate	
Fields	
+ requiredRights : number = 1	Stores the minimum authorisation rights for the guard

AuthGuard extends BaseGuard implements CanActivate	
Methods	
+ canActivate() : boolean	Returns a boolean of whether a user is logged in. If true, navigation continues. If false, navigation is cancelled.

DeviceGuard implements CanActivate	
Fields	
- router : Router	Injects the RouterService for “navigation among views”*
- device : DeviceService	Injects the DeviceService to retrieve information about the user’s device
Methods	
+ constructor(router: Router, device: DeviceService)	Instantiates an object of this class and injects the passed dependencies
+ canActivate( route: ActivatedRouteSnapshot, state: RouterStateSnapshot ) : boolean	Returns a boolean of whether a user’s device is suitable for the route being accessed. If true, navigation continues. If false, navigation is cancelled.

All persistent data for the system will be stored on a centralised database comprising 7 relations.

## USERS

Stores the identification and rights of every user on the system

	Name	Data Type	Unsigned	Auto Increment	Not Null	Default	Collation	Comment
<input checked="" type="checkbox"/>	user_id	SMALLINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
<input type="checkbox"/>	rights	ENUM('Students','Staff','Admins')	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		utf8mb4_unicode_ci	
<input type="checkbox"/>	first_name	CHAR(40)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		utf8mb4_unicode_ci	
<input type="checkbox"/>	surname	CHAR(40)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		utf8mb4_unicode_ci	
<input type="checkbox"/>	username	CHAR(8)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL	utf8mb4_unicode_ci	typically the first 3 letters from firstname and surname
<input type="checkbox"/>	password	CHAR(62)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL	utf8mb4_unicode_ci	bcrypt hash of password

```

CREATE TABLE pat_sandbox.users (
  user_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  rights ENUM('Students','Staff','Admins') NOT NULL,
  first_name CHAR(40) NOT NULL,
  surname CHAR(40) NOT NULL,
  username CHAR(8) DEFAULT NULL,
  password CHAR(62) DEFAULT NULL,
  PRIMARY KEY (user_id)
)
ENGINE = INNODB,
AUTO_INCREMENT = 77,
AVG_ROW_LENGTH = 356,
CHARACTER SET utf8mb4,
COLLATE utf8mb4_unicode_ci;


ALTER TABLE pat_sandbox.users
  ADD UNIQUE INDEX username(username);

```

user_id UNSIGNED SMALLINT	rights ENUM	first_name CHAR(40)	surname CHAR(40)	username CHAR(8)	password CHAR(62)
1	Staff	Glennie	Murrish	gmurri	\$2y\$12\$h7.4LU3ZQd09NLVrVRKD8.5xfXnqo5HbFFSIIRBV3iR.Iy9zMPdu
2	Students	Bone	Crocket	boncro	\$2y\$12\$h7.4LU3ZQd09NLVrVRKD8.5xfXnqo5HbFFSIIRBV3iR.Iy9zMPdu
3	Students	Vivi	Canty	vivcan	\$2y\$12\$h7.4LU3ZQd09NLVrVRKD8.5xfXnqo5HbFFSIIRBV3iR.Iy9zMPdu
4	Staff	Reyna	Kidds	rkidds	\$2y\$12\$h7.4LU3ZQd09NLVrVRKD8.5xfXnqo5HbFFSIIRBV3iR.Iy9zMPdu
5	Staff	Paxton	Braidley	pbraid	\$2y\$12\$h7.4LU3ZQd09NLVrVRKD8.5xfXnqo5HbFFSIIRBV3iR.Iy9zMPdu
6	Admins	Candace	Peach	cpeach	\$2a\$12\$zIrPEPbXGMqmbfDwxCyAeZ94dZxwkyUfquja330Y/rwseAvW2YX6
7	Students	Jenine	Holdron	jenhol	\$2y\$12\$h7.4LU3ZQd09NLVrVRKD8.5xfXnqo5HbFFSIIRBV3iR.Iy9zMPdu
8	Admins	Alf	Roddie	aroddi	\$2y\$12\$h7.4LU3ZQd09NLVrVRKD8.5xfXnqo5HbFFSIIRBV3iR.Iy9zMPdu
9	Students	Silvano	Deegan	sildee	\$2y\$12\$h7.4LU3ZQd09NLVrVRKD8.5xfXnqo5HbFFSIIRBV3iR.Iy9zMPdu
10	Students	Twyla	Colrein	twycol	\$2y\$12\$h7.4LU3ZQd09NLVrVRKD8.5xfXnqo5HbFFSIIRBV3iR.Iy9zMPdu

## STUDENTS

Stores the Administration Number of Student users

	Name	Data Type	Unsigned	Auto Increment	Not Null
<input checked="" type="checkbox"/>	user_id	SMALLINT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	admin_no	SMALLINT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

```
CREATE TABLE pat_sandbox.students (  
  user_id SMALLINT UNSIGNED NOT NULL,  
  admin_no SMALLINT UNSIGNED NOT NULL,  
  PRIMARY KEY (user_id)  
)  
ENGINE = INNODB,  
AVG_ROW_LENGTH = 431,  
CHARACTER SET utf8mb4,  
COLLATE utf8mb4_unicode_ci;  
  
ALTER TABLE pat_sandbox.students  
  ADD UNIQUE INDEX UK_students_admin_no(admin_no);  
  
ALTER TABLE pat_sandbox.students  
  ADD CONSTRAINT FK_students_user_id FOREIGN KEY (user_id)  
    REFERENCES pat_sandbox.users(user_id);
```

user_id UNSIGNED SMALLINT	admin_no UNSIGNED SMALLINT
10	1493
26	1504
73	1527
21	1529
22	1530

## NOTICES

Stores the contents and attributes of notices

✓	Name	Data Type	Unsigned	Auto Increment	Not Null	Default	Collation	Comment
<input checked="" type="checkbox"/>	notice_id	SMALLINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			Assumes 2 weeks averaging ~20 notices per day
<input type="checkbox"/>	user_id	SMALLINT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>			User who requested the notice
<input type="checkbox"/>	publish_date	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>			
<input type="checkbox"/>	display_date	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2020-02-25		
<input type="checkbox"/>	grouping_id	TINYINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL		
<input type="checkbox"/>	title	CHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		utf8mb4_unicode_ci	
<input type="checkbox"/>	description	VARCHAR(1000)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL	utf8mb4_unicode_ci	
<input type="checkbox"/>	approval_status	ENUM('Pending','Denied','Approved')	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Pending	utf8mb4_unicode_ci	

```
CREATE TABLE pat_sandbox.notices (
  notice_id SMALLINT NOT NULL AUTO_INCREMENT,
  user_id SMALLINT UNSIGNED NOT NULL,
  publish_date DATETIME NOT NULL,
  display_date DATE NOT NULL DEFAULT '2020-02-25',
  grouping_id TINYINT DEFAULT NULL,
  title CHAR(100) NOT NULL,
  description VARCHAR(1000) DEFAULT NULL,
  approval_status ENUM('Pending','Denied','Approved') NOT NULL DEFAULT 'Pending',
  PRIMARY KEY (notice_id)
)
ENGINE = INNODB,
AUTO_INCREMENT = 60,
AVG_ROW_LENGTH = 455,
CHARACTER SET utf8mb4,
COLLATE utf8mb4_unicode_ci;

ALTER TABLE pat_sandbox.notices
  ADD CONSTRAINT FK_notices_grouping_id FOREIGN KEY (grouping_id)
    REFERENCES pat_sandbox.groupings(grouping_id);

ALTER TABLE pat_sandbox.notices
  ADD CONSTRAINT FK_notices_user_id FOREIGN KEY (user_id)
    REFERENCES pat_sandbox.users(user_id);
```


notice_id SMALLINT	user_id UNSIGNED SMALLINT	publish_date DATETIME	display_date DATE	grouping_id TINYINT	title CHAR(100)	description VARCHAR(1000)	approval_status ENUM
1	1	2020/02/05 ...	2020/02/21	1	University Visits	Who: Crimson...	Approved
3	8	2020/02/20 ...	2020/02/21	3	Lost	An apple watc...	Pending
4	6	2020/01/21 ...	2020/02/21	4	Chapel Choir	Choir worksho...	Denied
5	6	2020/02/19 ...	2020/02/21	3	Lost	school bag bel...	Pending
6	1	2020/01/16 ...	2020/02/21	5	Gym Times	05h30 - 06h15	Denied
7	4	2020/02/20 ...	2020/02/21	6	Debating Society	meeting on Su...	Pending
8	5	2020/02/14 ...	2020/02/21	7	Snell Society	there is a Snel...	Pending
9	5	2020/02/13 ...	2020/02/21	8	Investment Club	Meeting in the...	Pending
10	8	2020/02/20 ...	2020/02/21	-4	Physio with Kir...	14h00 I Parso...	Pending



## GROUPINGS

Stores the name of each notification group and the section under which it falls

Negative numbered grouping\_id's belong to generic groups, facilitating a smoother transition from the analogue to the digital system


	Name	Data Type	Unsigned	Auto Increment	Not Null	Default	Collation	Comment
<input checked="" type="checkbox"/>	grouping_id	TINYINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
<input type="checkbox"/>	section_id	TINYINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL		based on the 6 currently used sections with leeway for expansion
<input type="checkbox"/>	name	CHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL	utf8mb4_unicode_ci	

```
CREATE TABLE pat_sandbox.groupings (  
  grouping_id TINYINT NOT NULL AUTO_INCREMENT,  
  section_id TINYINT DEFAULT NULL,  
  name CHAR(50) DEFAULT NULL,  
  PRIMARY KEY (grouping_id)  
)  
ENGINE = INNODB,  
AUTO_INCREMENT = 35,  
AVG_ROW_LENGTH = 744,  
CHARACTER SET utf8mb4,  
COLLATE utf8mb4_unicode_ci;  
  
ALTER TABLE pat_sandbox.groupings  
  ADD CONSTRAINT FK_groupings_section_id FOREIGN KEY (section_id)  
  REFERENCES pat_sandbox.sections(section_id);
```

grouping_id TINYINT	section_id TINYINT	name CHAR(50)
-6	6	To See Staff
-5	5	Restrictions
-4	4	Sanatorium
-3	3	Clubs and Societies
-2	2	Sport
-1	1	General Notices
1	1	Uni
3	1	Lost and Found
4	3	Chapel Choir
5	2	Gym Times

## SECTIONS

Stores the name of each section


	Name	Data Type	Unsigned	Auto Increment	Not Null	Default	Collation
<input checked="" type="checkbox"/>	section_id	TINYINT	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
<input type="checkbox"/>	name	CHAR(30)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL	utf8mb4_unicode_ci
<input type="checkbox"/>	sort_order	TINYINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL	

```
CREATE TABLE pat_sandbox.sections (  
  section_id TINYINT NOT NULL,  
  name CHAR(30) DEFAULT NULL,  
  sort_order TINYINT DEFAULT NULL,  
  PRIMARY KEY (section_id)  
)  
ENGINE = INNODB,  
AVG_ROW_LENGTH = 2730,  
CHARACTER SET utf8mb4,  
COLLATE utf8mb4_unicode_ci;
```

section_id TINYINT	name CHAR(30)	sort_order TINYINT
1	General Notices	2
2	Sport	3
3	Clubs and Societies	4
4	Sanatorium	5
5	Restrictions	6
6	To See Staff	1

## SUBSCRIPTIONS

Stores the subscriptions of each user to notification groups

	Name	Data Type	Unsigned	Auto Increment	Not Null
<input checked="" type="checkbox"/>	user_id	SMALLINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	grouping_id	TINYINT	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

```
CREATE TABLE pat_sandbox.subscriptions (  
  user_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  grouping_id TINYINT NOT NULL,  
  PRIMARY KEY (user_id, grouping_id)  
)  
ENGINE = INNODB,  
AUTO_INCREMENT = 74,  
AVG_ROW_LENGTH = 348,  
CHARACTER SET utf8mb4,  
COLLATE utf8mb4_unicode_ci;  
  
ALTER TABLE pat_sandbox.subscriptions  
  ADD CONSTRAINT FK_subscriptions_user_id FOREIGN KEY (user_id)  
    REFERENCES pat_sandbox.users(user_id);
```

user_id UNSIGNED SMALLINT	grouping_id TINYINT
1	1
3	12
4	1
4	10
5	9
6	1
6	4
6	5
6	7
6	12

## MENTIONS

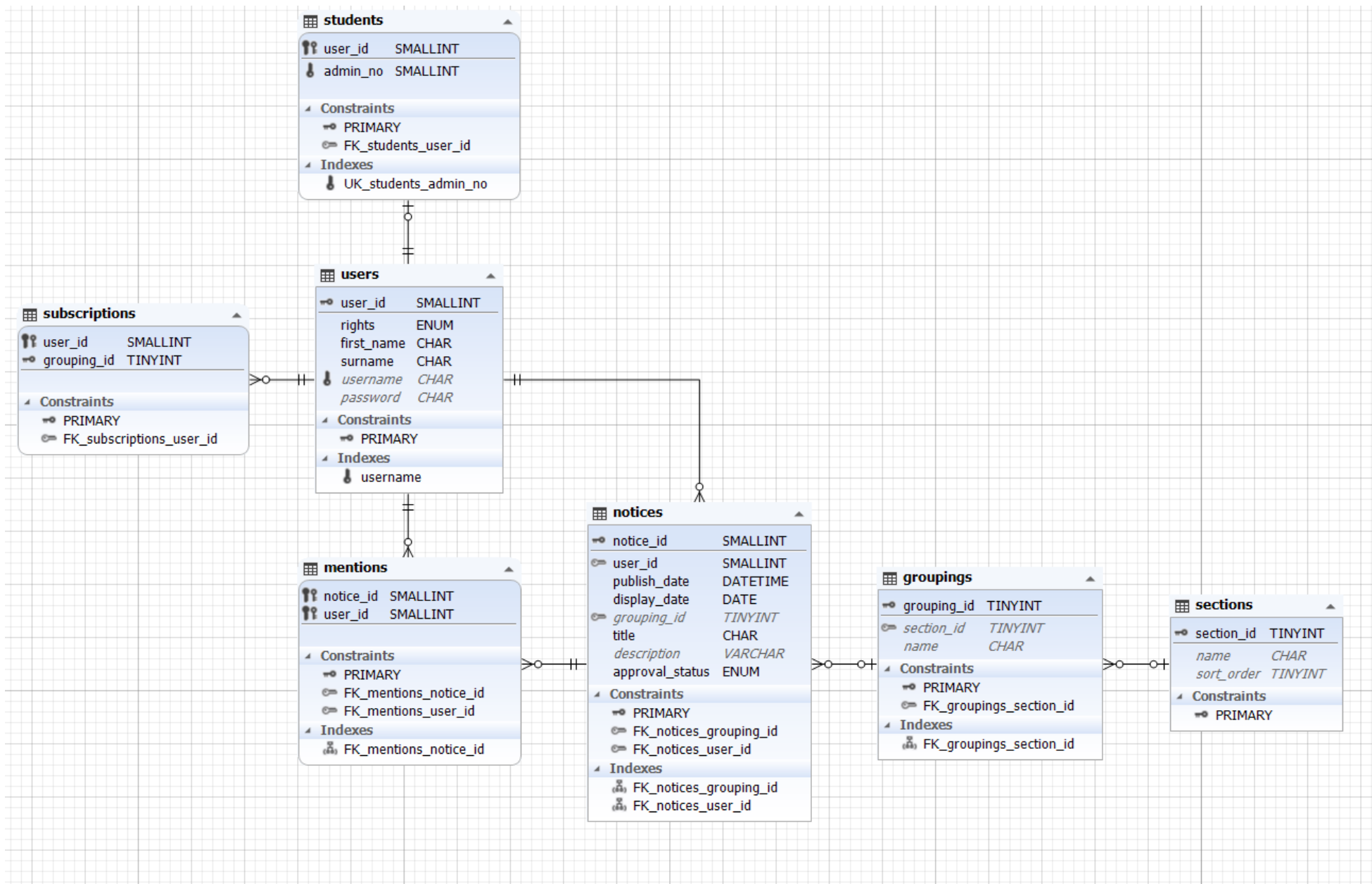
Stores the students who are tagged in particular notices

	Name	Data Type	Unsigned	Auto Increment	Not Null
<input checked="" type="checkbox"/>	notice_id	SMALLINT	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	user_id	SMALLINT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

```
CREATE TABLE pat_sandbox.mentions (  
  notice_id SMALLINT NOT NULL,  
  user_id SMALLINT UNSIGNED NOT NULL,  
  PRIMARY KEY (user_id, notice_id)  
)  
ENGINE = INNODB,  
AVG_ROW_LENGTH = 431,  
CHARACTER SET utf8mb4,  
COLLATE utf8mb4_unicode_ci;  
  
ALTER TABLE pat_sandbox.mentions  
  ADD CONSTRAINT FK_mentions_notice_id FOREIGN KEY (notice_id)  
    REFERENCES pat_sandbox.notices(notice_id);  
  
ALTER TABLE pat_sandbox.mentions  
  ADD CONSTRAINT FK_mentions_user_id FOREIGN KEY (user_id)  
    REFERENCES pat_sandbox.users(user_id);
```

notice_id SMALLINT	user_id UNSIGNED SMALLINT
5	11
10	12
10	13
11	17
11	18
12	20
12	21
13	22
13	23
14	26
14	27

## OVERVIEW AND RELATIONSHIPS



Rather than text files or a spreadsheet, I have opted for a relational database managed by MySQL of the owing to its support of the CRUD functions underpinning most of my application's procedures. In addition to offering the simplest implementation of the relationships between my tables compared to the other forms of storage, an RDBMS such as this will accommodate the large amounts of data on my application's system. This storage option boats another advantage in its data integrity protection features (such as constraints and indices).

I have structured my tables according to 3NF, normalising them to limit data redundancy and prevent anomalies.