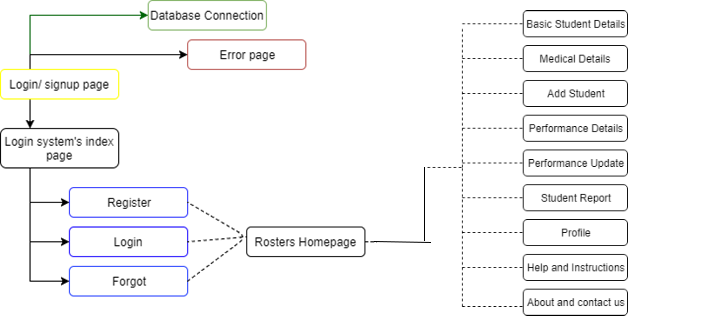
1. Page Hierarchy

Figure : Page hierarchy [[[1]](#footnote-1)] [[[2]](#footnote-2)]

1. Flowcharts for logic of data flow and processing

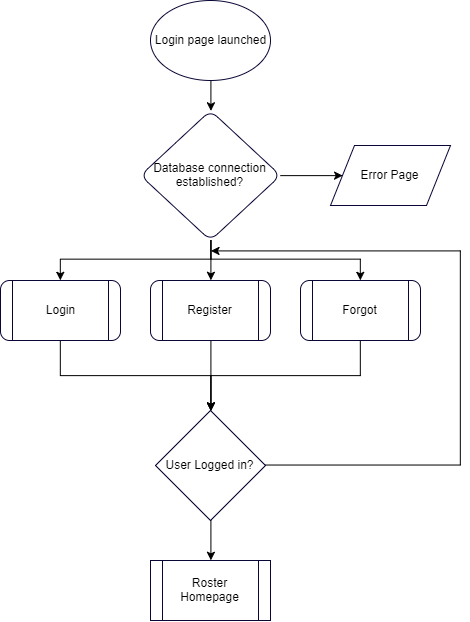


Figure : Flowchart of Logic flow Of website

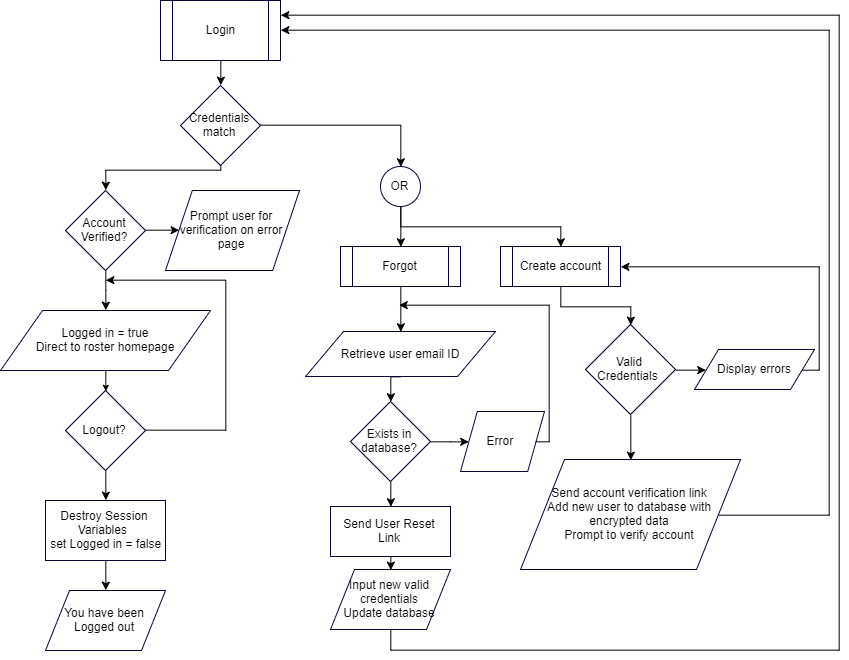


Figure : Logic Flowchart for subroutines: “Login”, “forgot”, and “register”

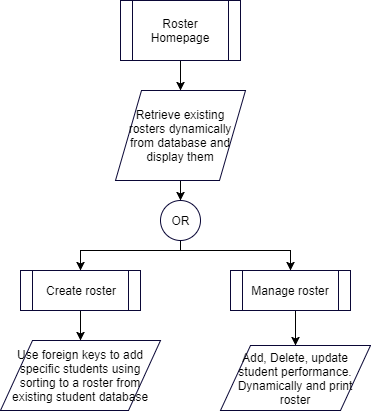


Figure :Logic flowchart for roster page

2.00 Database Structures

2.01

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Type | Data Type | Description | Validation | Verification | Reason NA |
| id | Integer | Int | Auto increment, serial number, and primary key | NA | NA | ID would be auto generated by the machine |
| first\_name | Text | Varchar | First name, entered by user | Text only | NA | User will be able to change it later on. Verification is not required since only a few people have access (teachers and school administration) |
| last\_name | Text | Varchar | Last name, entered by user | Text only | NA |
| email | Text, Symbols, Integers | Varchar | Email ID, entered by user | Email validation.  During login: does the entered email on login page match in database | Duplicate in database: Accounts? | NA |
| password | Text, Symbols, Integers | Varchar | Password, entered by user and encrypted into integers and text | More than 5 characters | During login: match the entered password’s hash version to that of in database | NA |
| hash | Integers and Text | Varchar | Hash, generated randomly based on password, md5 | NA | NA | Auto generated based on password |
| active | Binary | Tinyint(1) | Auto derived, binary (user registered via email: yes = 1, no = -1 | NA | NA | Auto assigned by deriving if user is verified by email id or not |

*Table 2.01*

2..02

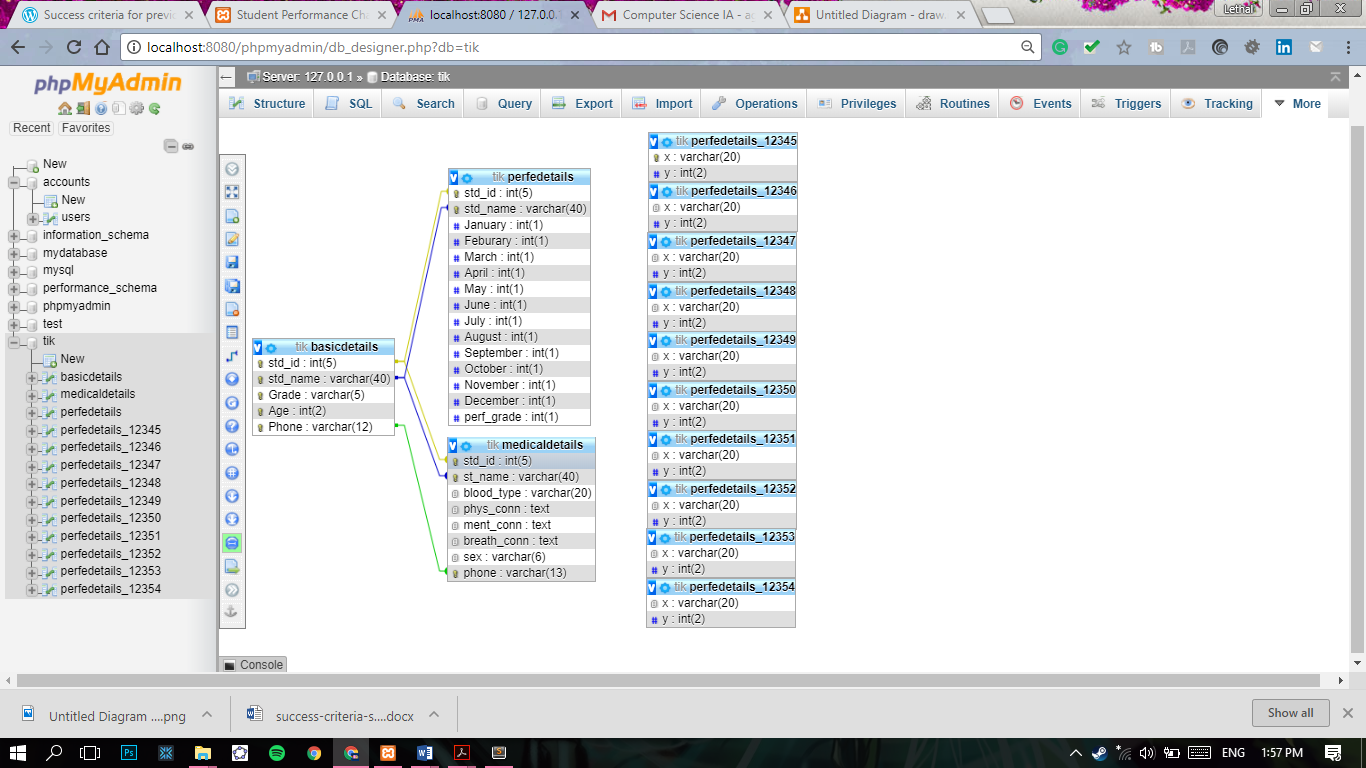
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Type | Data Type | Description | Validation | Verification | Reason NA |
| std\_id | Integers | Integer | Primary key, 5-digit id, entered by user | Primary key, error generated if duplicate, integers only | NA | User will be able to change or edit student’s ID |
| std\_name | Text | Varchar | Student name, entered by user | Text only | NA | User will be able to change student’s name |
| Grade | Integers, symbols, text | Varchar | Student grade, entered by user | NA | NA | User will be able to change student’s grade |
| Age(dob) | Integer | Integer | Student age, entered by user | Integers only | NA | User will be able to change student’s age |
| Phone | Integers, symbols | Varchar | Student phone number, entered by user | 12 digits maximum | NA | User will be able to change phone |

*Table 2.02*

2.03

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Type | Data Type | Description | Validation | Verification | Reason NA |
| Std\_id | Integers | Int | Foreign Key and composite with date, 5 digits retrieved from database | NA | NA | Already validated and verified |
| Std\_name | Text and symbols | Varchar | Student name, entered by user | NA | NA | Already Validated and verified |
| Perf\_Grade[[3]](#footnote-3) (for each month Jan to Dec) | Positive real numbers  0 to 10 | Int | Student grade entered by user | Grade is in range 0 to 10 | NA | Entered by user and cannot be properly assessed to ascertain validity |

2.04

[[4]](#footnote-4)

3.10. CSS styling [[5]](#footnote-5)

Styles consistent throughout the pages: with reference to appendix images

3.11. Default Styles

Container

Left: 25%;

Margins top: -3.5%

Margin Bottom: 20%;

Body

Font: Google fonts- Robboto and Dosis

Note: A high contrasting and bright yet earthy colour pallet was chosen because the swimming pool is indoors but there are windows huge enough that make it harder to see the screen

3.12. Side/ Left navbar (vertical)

Width: 20% of border box (browser window)

Position: fixed

Background colour: #1c2b36;

Always visible

Text align center

3.12. Custom scrollbar

Bootstrap webkit scrollbar

Note: A custom scrollbar was chosen because most teachers were using old and legacy smartphone systems due to which some browser’s scrollbars were not responsive or “hard to navigate around with”

3,14 Table styles

Row colour: #ffffff

Background colour: #f6f8f8

Separated rows with shadows

4.00 Pseudocode for algorithms to be used

4.01 Pseudocode for establishing database connections

|  |
| --- |
| initialize variables: Host, user, pass, and db and assign values  query = new mysqli(Host, user, pass, db)  run mysqli query or die and prompt user |

4.02 Pseudocode for submission of login/ registration data

|  |
| --- |
| Establish server connection (relay to error page if connection fails)  Select database accounts  Initiate session variables (if submit is pressed)  variables = data entered by the user  use hash and salt to escape variables to prevent any SQL injections //refer to the pseudocode in 4.04  If error is raised: prompt user  Else update/ create/ delete variables in tables  On logout: destroy session variables, end connection with database |

4.03 Pseudocode to update forgotten credentials

Establish server connection (relay to error page if connection fails)

Select database accounts

Initiate session variables (if submit is pressed)

variables = data entered by the user

use hash and salt to escape variables to prevent any SQL injections //refer to the pseudocode in 4.04

If error is raised: prompt user

Else if check email address matches to any of that in the database

Send credential-reset email

endif

if reset initiated

verify user's passwords

If errors: prompt user

else: update database with relevant data

prompt user to login

On logout: destroy session variables, end connection with database

4.04 Pseudocode for “Hash and salt” http://php.net/manual/en/password.constants.php

|  |
| --- |
| Variable pass = md5 encryption of (data entered)  And add a random salt by using password bcrypt |

4.05 Pseudocode for dynamic entry of database on the website

|  |
| --- |
| Retrieve information from database in objects  using AJAX retrieve all the data into variables dynamically  repeat while table.hasnext()  Retrieve each row  Print row |

4.05 Pseudocode for updating database

|  |
| --- |
| Session start  establish database connection  escape all strings/ data student id and new grade with any changes to old grade and a particular day  using sql: where id = id entered and month= month update perf grade to grade entered by the user  “flush” all session variables  close sql query  4.06 Pseudocode for constructing dynamic charts[[6]](#footnote-6) |

foreach($result as $row){

array\_push($dataPoints, array("x"=> $row->x, "y"=> $row->y

}

data: [{

type: "area",

dataPoints: <?php echo json\_encode($dataPoints, JSON\_NUMERIC\_CHECK); ?>

}]

4.07 pseudocode for static chart

$dataPoints = array(

array("label"=> 'January', "y"=> 2),

array("label"=> 'Feburary', "y"=> 5

………

);

data: [{

type: "area",

dataPoints: <?php echo json\_encode($dataPoints, JSON\_NUMERIC\_CHECK); ?>

}]

Chart.render

Use resource canvasjs

5.00 Test Plan

|  |  |  |
| --- | --- | --- |
| Test Type | Nature Of test | Example |
| Submission button – login page | If entered credentials are right, relay user to the roster page; else relay user to error page | Email id = [xyz@xyz.com](mailto:xyz@xyz.com) and pass =123  But user entered  Xyz.com and pass 123  Prompt: invalid email: missing “@”  Enter correct information and user is relayed to the home page. |
| Submission button - register | If entered credentials are valid, relay user to the logout page and prompt to verify email address by the link sent to the registered email. Check if email is sent and user’s account is activated. | Register:  Email = [xyz@xyz.com](mailto:xyz@xyz.com)  Pass= 123  Name = xyz  …. Etc… and email with link to verify is received |
| Submission button - logout | Check if user is logged out (all session variables are destroyed) | Logout and check if user is still logged in the system |
| Database connection | Check if database is connected | Retrieve data from database |
| Data entry testing | Check if data is entered into the right fields and database is accepting data. | Console.log out any errors generated and check database for data entered |
| Cross platform compatibility- responsive design test | Compatibility test | Running website on mobile devices and different screen sizes |
| Cross browser compatibility | Compatibility test | Running website on different browsers |
| Testing for time taken to load websites and pages | Efficiency test | Run website by throttling internet speeds |
| PDF generation testing | Correct detail generation | The PDF has correct and relevant details |
| Grading testing | Student grades are properly calculated and display their progress/ performance | A student who qualifies for the districts must have a score greater than that of 6.5 |
| JavaScript testing | JavaScript is properly working on all pages. | Check each page for any animations and JavaScript errors |
| CSS testing | Check if css is working across all pages | Css is easy to use and makes the website intuitive to use |
| UI/ UX testing | Check if the user is able to navigate through the UI/ UX with ease and is visible even with direct sunlight on screen | Website increase efficiency and helps the teacher with ease of use. |

1. Refer to Appendix: 1 Question: 7 [↑](#footnote-ref-1)
2. Refer to appendix: 2 *(to understand the design of the website)* [↑](#footnote-ref-2)
3. Refer to Appendix: 1 Question: 9 [↑](#footnote-ref-3)
4. Each perfedetails\_##### table represents performance grades for each student. ##### represents student ID, therefore all these tables have a 1 to 1 relationship with std\_id in basic details table [↑](#footnote-ref-4)
5. Refer to Appendix: 1 Question: 7; and Refer to Appendix: 2 [↑](#footnote-ref-5)
6. This code is included in the project: Appendix :Student chart front end [↑](#footnote-ref-6)