**Initial Proposal**

**As a business owner I would like to propose a system that fair the school management system could be automated and scripted as below . And it is containing the following structures like student class and this class to subject and subject to a teacher and class teacher and an branch. And this will contain the data completely manipulated by using this tool away how the student is associated with the student and the class 10 class teacher and the subject and the branch.**

**Relational Database Design Process:**

**Using the tableau have created a simple data model where student can able to access the complete database to the database systems that will be containing the five different entities that are students teachers class teachers subject teachers. Please find below screenshot that defines the data model of this school management system**

Diagram

Description automatically generated

**Data Sources:**

**Generally As a Business owner I am tinking to have this complete information need to be drafted in the tables using the application. So generally we will use the form elements. In some special cases Such as bulk upload of subjects students and maping the students and teachers could also be uploaded in bulk in that aspect we need to support the JSOn /CSV data sources.**

**Data Definition Language Scripts**

Diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

Scripts:

CREATE TABLE branch (

    name varchar(128) NOT NULL,

    description text NOT NULL,

    status enum('y','n') NOT NULL,

    id int(4) NOT NULL AUTO\_INCREMENT,

    CONSTRAINT branch\_pk PRIMARY KEY (id)

) ENGINE InnoDB CHARACTER SET utf8mb4;

-- Table: class

CREATE TABLE class (

    name varchar(11) NOT NULL,

    section varchar(11) NOT NULL,

    id int(11) NOT NULL AUTO\_INCREMENT,

    CONSTRAINT class\_pk PRIMARY KEY (id)

) ENGINE InnoDB CHARACTER SET utf8mb4;

-- Table: students

CREATE TABLE students (

    name varchar(128) NOT NULL,

    classid int(4) NOT NULL,

    classteacher int(4) NOT NULL,

    id int(4) NOT NULL AUTO\_INCREMENT,

    CONSTRAINT students\_pk PRIMARY KEY (id)

) ENGINE InnoDB CHARACTER SET utf8mb4;

CREATE INDEX name ON students (name);

CREATE INDEX classid ON students (classid);

CREATE INDEX classteacher ON students (classteacher);

-- Table: subjects

CREATE TABLE subjects (

    name varchar(11) NOT NULL,

    id int(11) NOT NULL AUTO\_INCREMENT,

    CONSTRAINT subjects\_pk PRIMARY KEY (id)

) ENGINE InnoDB CHARACTER SET utf8mb4;

-- Table: teacher\_subjects

CREATE TABLE teacher\_subjects (

    teacherid int(11) NOT NULL,

    subjectid int(11) NOT NULL,

    classid int(11) NOT NULL,

    id int(11) NOT NULL AUTO\_INCREMENT,

    CONSTRAINT teacher\_subjects\_pk PRIMARY KEY (id)

) ENGINE InnoDB CHARACTER SET utf8mb4;

CREATE INDEX teacherid ON teacher\_subjects (teacherid);

CREATE INDEX subjectid ON teacher\_subjects (subjectid);

CREATE INDEX classid ON teacher\_subjects (classid);

-- Table: teachers

CREATE TABLE teachers (

    name varchar(128) NOT NULL,

    stuatus int(4) NOT NULL,

    id int(4) NOT NULL AUTO\_INCREMENT,

    CONSTRAINT teachers\_pk PRIMARY KEY (id)

) ENGINE InnoDB CHARACTER SET utf8mb4;

-- foreign keys

-- Reference: students\_ibfk\_1 (table: students)

ALTER TABLE students ADD CONSTRAINT students\_ibfk\_1 FOREIGN KEY students\_ibfk\_1 (classid)

    REFERENCES class (id)

    ON DELETE CASCADE

    ON UPDATE CASCADE;

-- Reference: students\_ibfk\_2 (table: students)

ALTER TABLE students ADD CONSTRAINT students\_ibfk\_2 FOREIGN KEY students\_ibfk\_2 (classteacher)

    REFERENCES teachers (id)

    ON DELETE CASCADE

    ON UPDATE CASCADE;

-- Reference: teacher\_subjects\_ibfk\_1 (table: teacher\_subjects)

ALTER TABLE teacher\_subjects ADD CONSTRAINT teacher\_subjects\_ibfk\_1 FOREIGN KEY teacher\_subjects\_ibfk\_1 (subjectid)

    REFERENCES subjects (id)

    ON DELETE CASCADE

    ON UPDATE CASCADE;

-- Reference: teacher\_subjects\_ibfk\_2 (table: teacher\_subjects)

ALTER TABLE teacher\_subjects ADD CONSTRAINT teacher\_subjects\_ibfk\_2 FOREIGN KEY teacher\_subjects\_ibfk\_2 (teacherid)

    REFERENCES teachers (id)

    ON DELETE CASCADE

    ON UPDATE CASCADE;

-- Reference: teacher\_subjects\_ibfk\_3 (table: teacher\_subjects)

ALTER TABLE teacher\_subjects ADD CONSTRAINT teacher\_subjects\_ibfk\_3 FOREIGN KEY teacher\_subjects\_ibfk\_3 (classid)

    REFERENCES class (id)

    ON DELETE CASCADE

    ON UPDATE CASCADE;

-- End of file.

Data Manipulations:

[INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO `students` (`id`, `name`, `classid`, `classteacher`) [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (NULL, 'XNAME', '2', '5'), (NULL, 'YNAME', '3', '3');

Graphical user interface, text, application, email

Description automatically generated

UPDATE `students` SET `name` = 'FNGM' WHERE `students`.`id` = 2;

Graphical user interface, text, application

Description automatically generated

UPDATE `students` SET `name` = 'YUGI' WHERE `students`.`id` = 1;

Graphical user interface, text, application

Description automatically generated

SELECT \* FROM `students`

Graphical user interface, text, application

Description automatically generated

SELECT s.\*,t.name teacher,sub.name as subject FROM `students` s

INNER JOIN teachers t ON s.classteacher=t.name

INNER JOIN teacher\_subjects ts ON ts.teacherid=t.id

INNER JOIN subjects sub ON sub.id=ts.subjectid

Graphical user interface, text, application, email

Description automatically generated

SELECT s.\*,t.name teacher,sub.name as subject FROM `students` s

LEFT JOIN teachers t ON s.classteacher=t.name

LEFT JOIN teacher\_subjects ts ON ts.teacherid=t.id

LEFT JOIN subjects sub ON sub.id=ts.subjectid

Graphical user interface, text, application, email

Description automatically generated

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM `teacher\_subjects`

[UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) `teacher\_subjects` [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) `classid` = '1' WHERE `teacher\_subjects`.`id` = 4;

Graphical user interface, application

Description automatically generated

SELECT s.\*,t.name teacher,sub.name as subject FROM `students` s

LEFT JOIN teachers t ON s.classteacher=t.name

LEFT JOIN teacher\_subjects ts ON ts.teacherid=t.id

LEFT JOIN subjects sub ON sub.id=ts.subjectid

**Indexes:**

**Indexses are most important in the database to make the complete database reliable and make the proper and seamless performance. By defining the Index we can define thee masa the refeecne table to other columns in the pilot tables so in the fk it is very important have indexes.**

CREATE INDEX teacherid ON teacher\_subjects (teacherid);

CREATE INDEX subjectid ON teacher\_subjects (subjectid);

CREATE INDEX classid ON teacher\_subjects (classid);

ALTER TABLE teacher\_subjects ADD CONSTRAINT teacher\_subjects\_ibfk\_1 FOREIGN KEY teacher\_subjects\_ibfk\_1 (subjectid)

    REFERENCES subjects (id)

    ON DELETE CASCADE

    ON UPDATE CASCADE;

[SHOW](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/show.html) indexes FROM teacher\_subjects

Graphical user interface, text, application, email

Description automatically generated

**Views:**

**View can be called as the virtual tables where we can specify the most resued columns in the multiple tables can be viewed as the single table . for example we have 4 different tables such as Studets teachers subjects and Branch all these are references for the student and teacher tables if we want vie all the coulms in a single table we can define view and can able to view the information.**

CREATE VIEW List

AS

SELECT s.\*,t.name teacher,sub.name as subject FROM `students` s

LEFT JOIN teachers t ON s.classteacher=t.name

LEFT JOIN teacher\_subjects ts ON ts.teacherid=t.id

LEFT JOIN subjects sub ON sub.id=ts.subjectid

Graphical user interface, text, application

Description automatically generated

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-view.html) [VIEW](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-view.html) Studentteacher AS [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) s.\*,t.name teacher FROM `students` s [LEFT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-functions.html#function_left) JOIN teachers t ON s.classteacher=t.name [LEFT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-functions.html#function_left) JOIN teacher\_subjects ts ON ts.teacherid=t.id

Graphical user interface, text, application, Word

Description automatically generated

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM `list`

Graphical user interface, text, application

Description automatically generated

Triggers:

Triggers is a mechanism where we will give the associated task after any kind of transaction such as Insert update or delete We can specify this with the simple trigger.

CREATE TRIGGER Del BEFORE INSERT ON subjects

Graphical user interface, text, application

Description automatically generated

Transactions:

Transactions is an very important feature in the mysql database were we can commit the complete set of transactions where we can commit the list of transactions or list of operation or linked operations. If we have specified in the chain of linked transactions they can be rolled bask if any transaction in that list is failed. Using the ROLLBACK keyword.

START TRANSACTION;

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM `teacher\_subjects`

[UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) `teacher\_subjects` [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) `classid` = '1' WHERE `teacher\_subjects`.`id` = 4;

ROLLBACK;

Graphical user interface, text, application

Description automatically generated

Locking:

Locking the database tables will grant the user to lock the user from being Writing or Reading permissions form the user to that particular database. It is just similar mechanisms where the permission for the direcotires.

[LOCK](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/lock-tables.html) [TABLES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/lock-tables.html) teacher\_subjects READ

Graphical user interface, text, application, Word

Description automatically generated

Backup :

Backing up the complete dadtabase is quite simple and secure if we using the linuz environments and MySQL cli. But in our case we are using the mariadb so we need to specify the dump loacationa dn then we have to run the sQl to backup the data. By defining the croon Job we can define the frequency to make the complete site backup timely. For security we can make the complete folder where the backups are being store are need to be secured by the folder secureity mechanism.

mysqldump --databases --user=root --password your\_db\_name > export\_into\_db.sql

Python:

import mysql.connector

dataBase = mysql.connector.connect(

host ="localhost",

user ="root",

passwd ="",

database='schooldb'

)

cursor = dataBase.cursor()

query = ("select \* from studentteacher")

cursor.execute(query)

#print(dataBase)

for (name) in cursor:

print(name)

# Disconnecting from the server

cursor.close()

dataBase.close()

Graphical user interface, application

Description automatically generated

PHP Application to search Students with teacher :

In order to have the application we will create a sample html file where the the simple form to get input data from the user the index.html file look like below:

<!DOCTYPE html>

<html>

  <head>

    <title>School DB search</title>

    <script>

    function ajsearch () {

      // (A) GET SEARCH TERM

      var data = new FormData();

      data.append("search", document.getElementById("search").value);

      data.append("ajax", 1);

      // (B) AJAX SEARCH REQUEST

      fetch("search.php", { method:"POST", body:data })

      .then(res => res.json()).then((results) => {

        var wrapper = document.getElementById("results");

        if (results.length > 0) {

          wrapper.innerHTML = "";

          for (let res of results) {

            let line = document.createElement("div");

            line.innerHTML = `${res["name"]} - ${res["teacher"]}`;

            wrapper.appendChild(line);

          }

        } else { wrapper.innerHTML = "No results found"; }

      });

      return false;

    }

    </script>

  </head>

  <body>

    <!-- (A) SEARCH FORM -->

    <form onsubmit="return ajsearch();">

      <h1>SEARCH FOR USERS</h1>

      <input type="text" id="search" required/>

      <input type="submit" value="Search"/>

    </form>

    <!-- (B) SEARCH RESULTS -->

    <div id="results"></div>

  </body>

</html>

Then Here we are using the ajax call where the request is to be flow through the ajax call and then the resultant data will be appended to the div where we specified.

<?php

// (A) DATABASE CONFIG - CHANGE TO YOUR OWN!

define("DB\_HOST", "localhost");

define("DB\_NAME", "schooldb");

define("DB\_CHARSET", "utf8");

define("DB\_USER", "root");

define("DB\_PASSWORD", "");

// (B) CONNECT TO DATABASE

try {

  $pdo = new PDO(

    "mysql:host=".DB\_HOST.";charset=".DB\_CHARSET.";dbname=".DB\_NAME,

    DB\_USER, DB\_PASSWORD, [

      PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

      PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC

    ]

  );

} catch (Exception $ex) { exit($ex->getMessage()); }

// (C) SEARCH

$stmt = $pdo->prepare("SELECT s.\*,t.name teacher FROM `students` s JOIN teachers t on s.classteacher=t.id WHERE s.name LIKE ? ");

$stmt->execute(["%".$\_POST["search"]."%"]);

$results = $stmt->fetchAll();

if (isset($\_POST["ajax"])) { echo json\_encode($results); }

Here is the sample screen

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with low confidence

If you observe the Inspect tools network tab a sample search request made and response being print on the same page without reload.

Graphical user interface, text, application

Description automatically generated

Prevention of the SQL Injection : Here we are using the Ajax call so we are in a vulnerable to the data loss because the data will be expose on the browser if you see the last screen shot the complete data is being exposed. So to being secured from the SQL injection I will use the Authentication Barier inorder to secure the complete data.