University of the Philippines

Guidance Office

Student Information Database and Risk Protective Assessment Survey

Tyrel Justin Dogup

Peter John Ramos

Enrico Baello Jr.

Joel Ivan Sarmiento

Preface

This is a documentation on the Computer Science 128 project by Tyrel Justin Dogup, Peter John Ramos, Enrico Baello Jr., and Joel Ivan Sarmiento. The project is a student information database and mental health survey requested by the Guidance office of the University of the Philippines – Baguio Campus.

Introduction

The project was made in accordance to the request of the Guidance office. The major part of the system aims to lessen manual labor for the staff and lessen paperwork for a greener and environmental-friendly campus. A minor part of the system aims to assess the mental health of students residing in the University of the Philippines – Baguio Campus in order to provide the proper guidance in case of a high risk assessment.

The system was made with CodeIgniter as a framework which uses an MVC pattern. CodeIgniter is a PHP framework with a very small footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications. The system was also built using AngularJS which helps in the interface and some functions in the system.

The major part of the system is a student information database. This was created to store the valuable student information the Guidance office needs like background information, educational background, etc. The tables in this section are editable so as to help in future additional information needed by the office.

The second function of the system is a mental health survey or risk protective assessment survey. This was created to assess a student’s mental health and the risks of him or her committing self-harm or any suicidal thoughts. This assessment aims to provide the Guidance office with the information it needs so as to provide a better advice on any student with his or her problems.

Glossary

Database - a structured set of data held in a computer, especially one that is accessible in various ways.

Administrator - a person responsible for running a business, organization, etc.

User Requirements Definition

**Non-functional Requirements**

Security – An administrator function has been added so that students or other users would not be able to access the student information page and mental health survey page.

Performance and Usability – Use of the system is made easier with a simple user interface with clear buttons and some shortcuts.

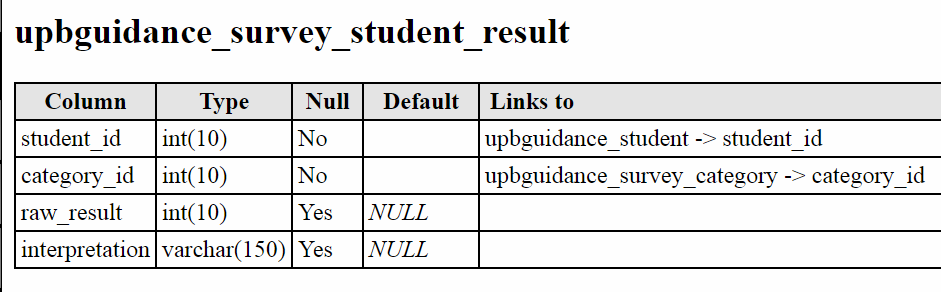
**Functional Requirements**

Administrator Privileges – This function was created to ensure the confidentiality of the information provided by the student in the Student Information Page.

Student Information Database – This function was created to store a student’s information. Default fields include background information, family data, educational background, and financial information. Student information can be added, deleted, updated, or printed.

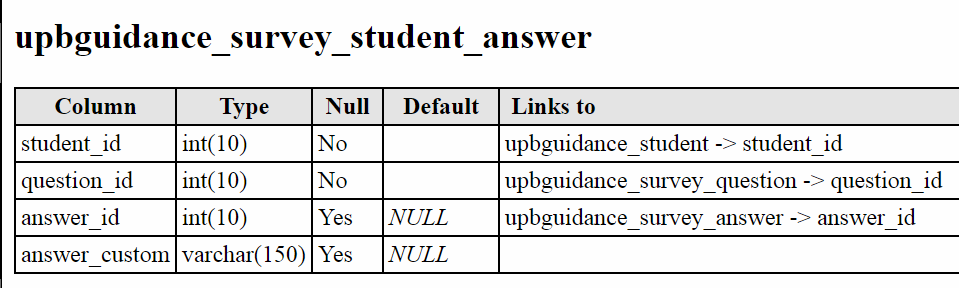
Risk Protective Assessment Survey – This function was created to help the Guidance office assess a student’s mental health and to better understand how to help these students overcome these problems.

System Architecture

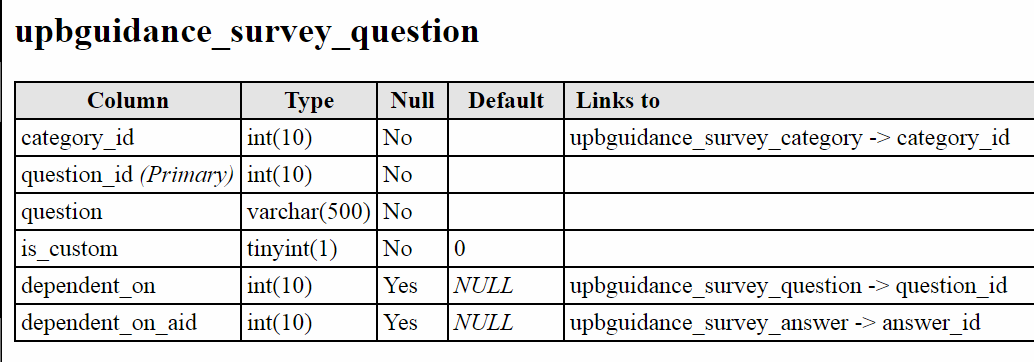


**Survey student results table**

This table stores the results and interpretation of the student’s answer to the risk protective assessment survey. This table is linked to the student id in the student table so as tell which result belongs to who. It is also linked to the category id in the survey category table in order to assess which result belongs to a certain category.

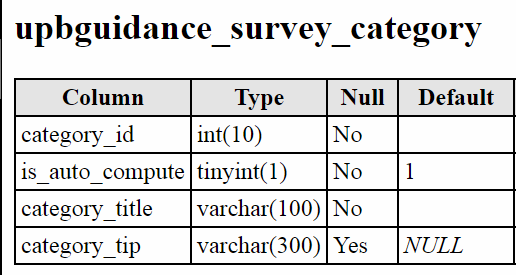


**Survey student answer table**

This table stores the answer of the student to a question in the Risk Protective Assessment Survey. This is linked to the student id of the student table so as to know which student answered a specific answer set. This table is also linked to the question id of the survey question table to check which question was answered. This table is also linked to the answer id of the survey answer table to determine what assessment would be given to the student’s answer.

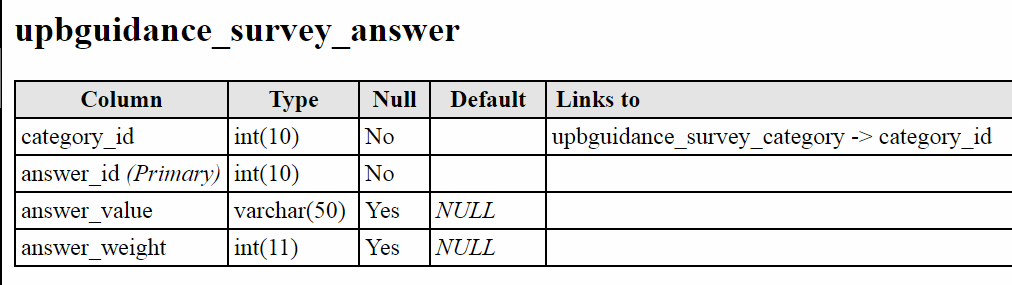
**Survey question table**

This table stores the questions in the Risk Protective Assessment Survey. This is linked to the category id of the survey category table to determine the category of a question. The ‘dependent on’ variable is linked to the question id of the survey question table for questions which can only be assessed depending on the question id it is linked to. The ‘dependent on aid’ variable is linked to the answer id of the survey answer since some questions can only be answered depending on the student’s answer on the previous question.



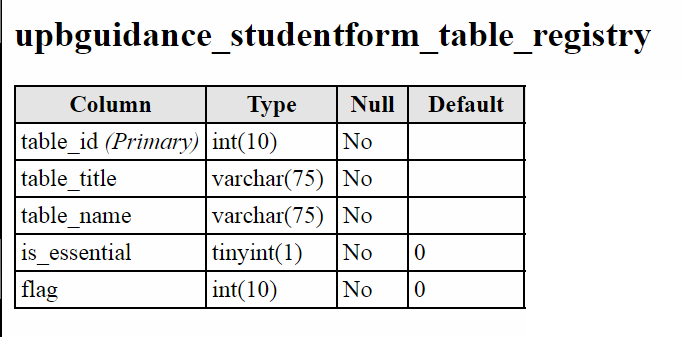
**Survey category table**

This table stores the different categories of the Risk Protective Assessment Survey.



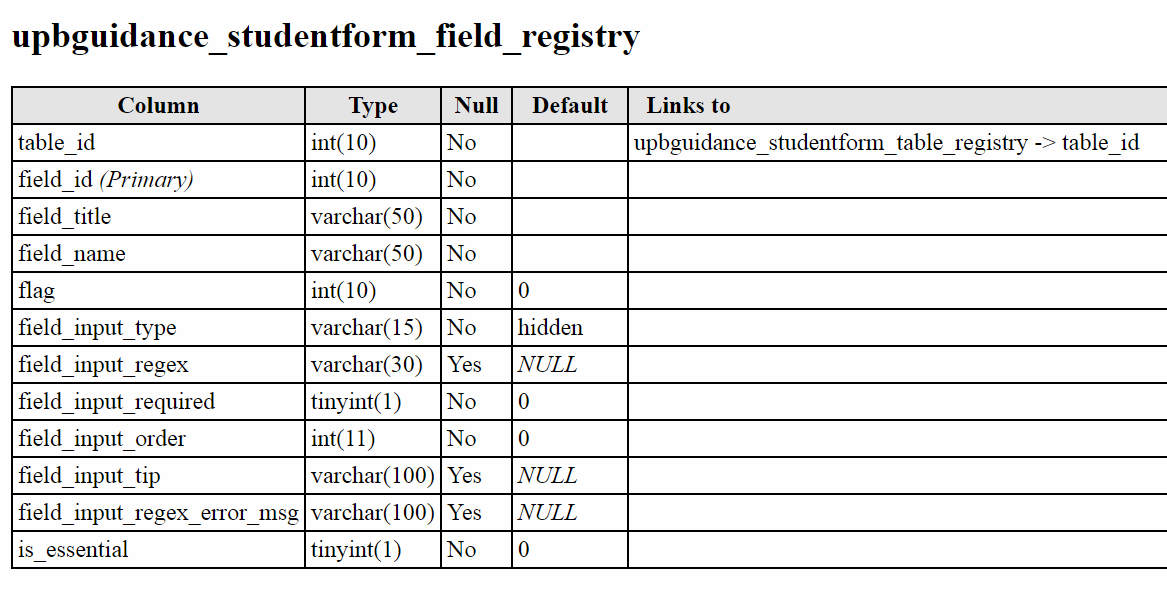
**Survey Answer table**

This table stores the answer key to the Risk Protective Assessment Survey. In truth there are no correct answers to the survey, what this table does is give a weight to a specific answer which is then interpreted at the survey student result table.



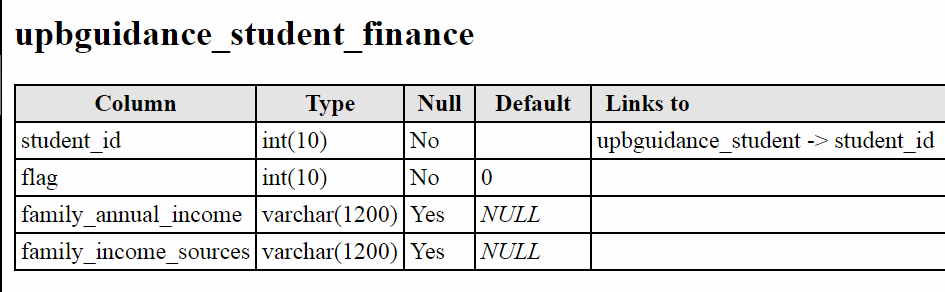
**Student form table registry table**

This table stores the information on newly created tables in the Student Information Page.



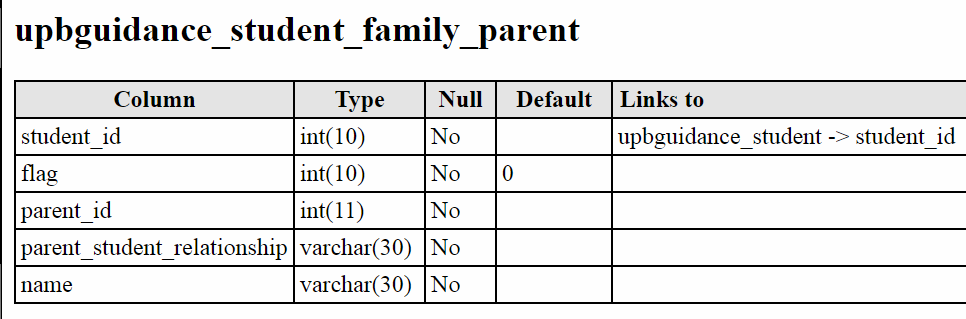
**Student form field registry table**

This table stores information on newly created fields in the Student Information Page. This is linked to the table id of the student form table registry table since all fields are under a table.



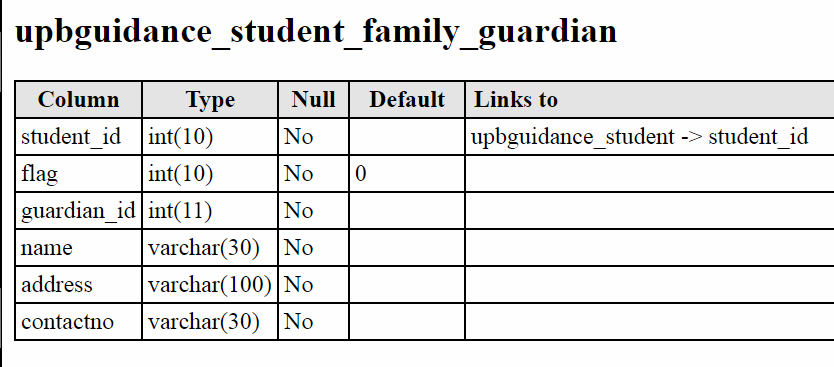
**Student finance table**

This stores the student’s financial information entered through the Students Information Page.



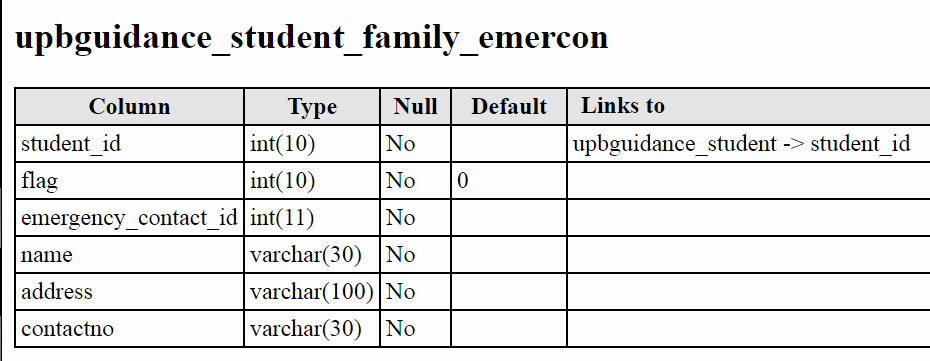
**Student family parent table**

This table stores the student’s parent’s information entered from the Student Information Page.



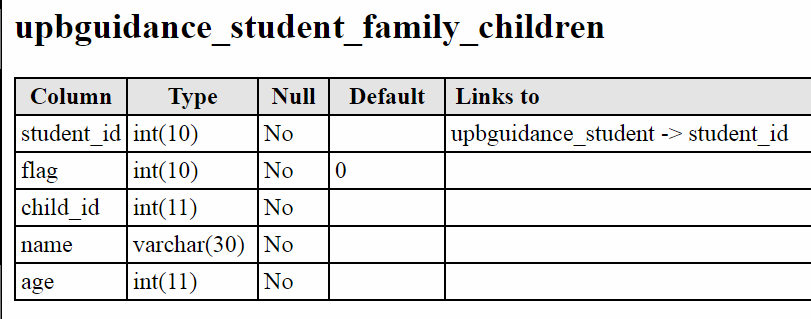
**Student family guardian table**

This table stores the student’s guardian’s information entered from the Student Information Page.



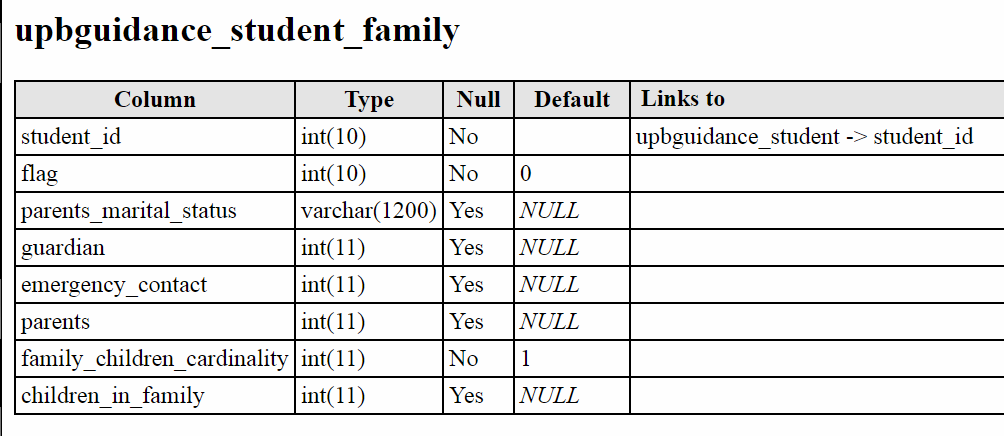
**Student family emercon table**

This table stores the student’s emergency contact information entered from the Student Information Page.



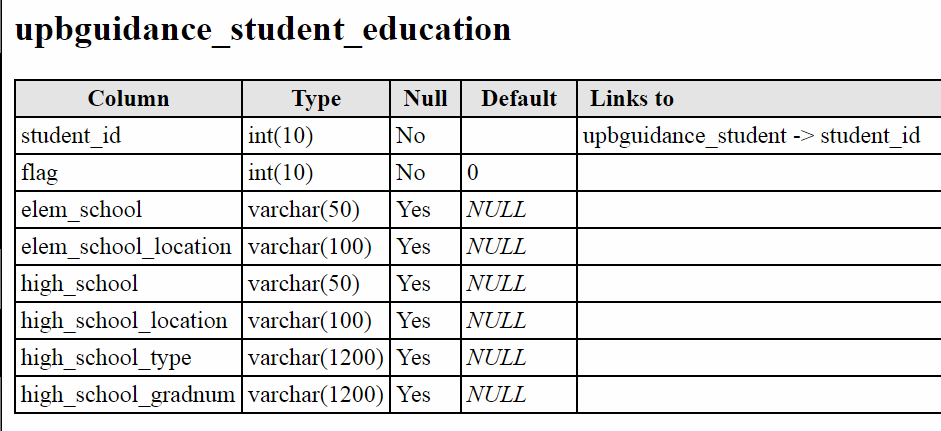
**Student family children table**

This table stores the student’s sibling’s information entered through the Student Information Page.



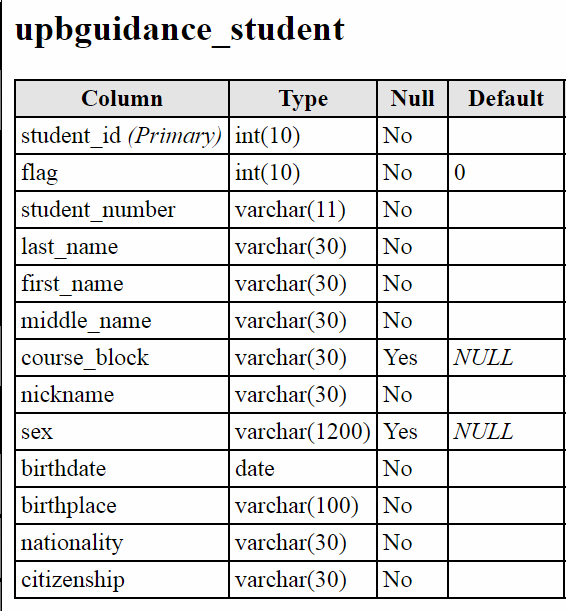
**Student family table**

This table helps in linking the student to his guardian, emergency contact, parents, and siblings.



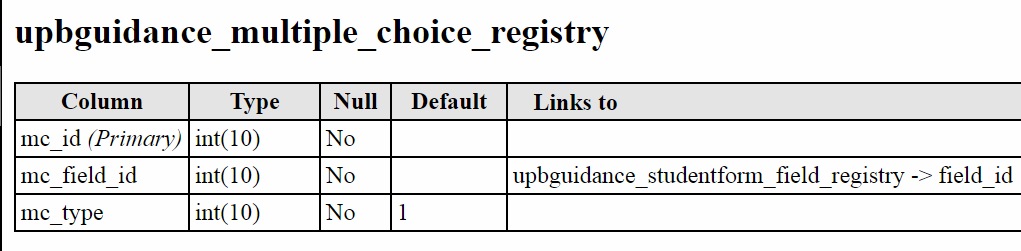
**Student education table**

This table stores the student’s educational background.



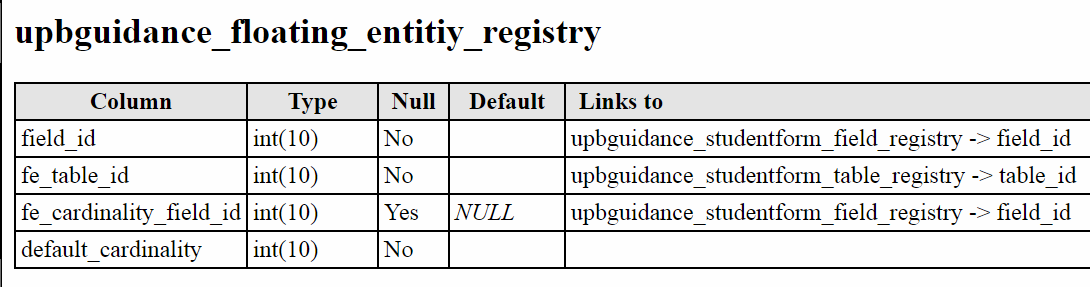
**Student table**

This table stores the student’s background information.



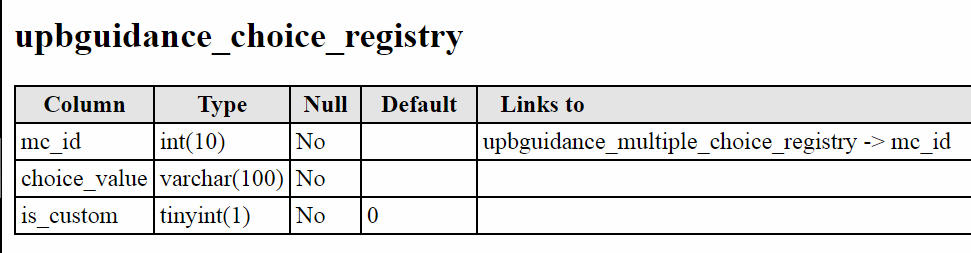
**Multiple choice registry table**

This table is linked to the field id of the student for field registry table and allows newly created fields to have multiple choices and allow storage of multiple answers for a single question.



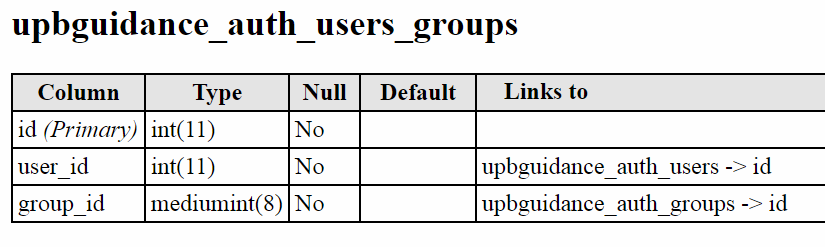
**Floating entity registry**

This table is linked to the field id of the student form field registry table. Floating entities are sub-tables that allow multiple instances under the same questions or fields.



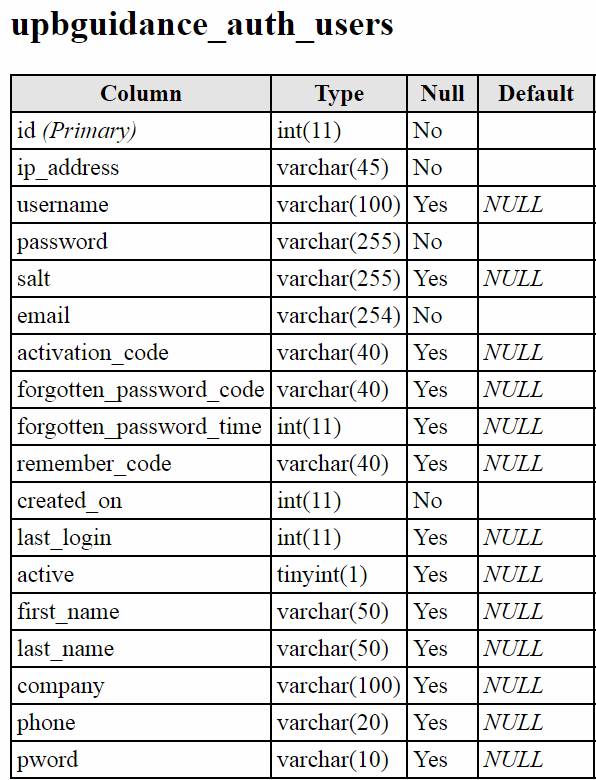
**Choice registry table**

Adds values to the choices of the multiple choice registry table.



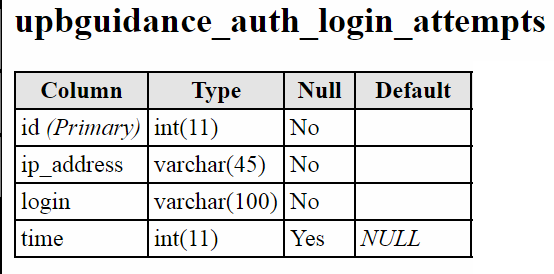
**Auth users groups table**

Stores the different groupings of login accounts.



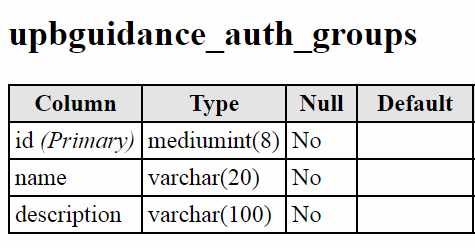
**Auth users table**

Stores the important information of the administrator.



**Auth login attempts table**

This table stores the count of how many times the user has attempted to login using the linked account and has failed to provide the correct password.



**Auth groups table**

This table stores the name and description of the different account groups.

System Models

The system uses the CodeIgniter framework which is a toolkit for people who build web applications using PHP. It made developing the project faster than writing from scratch by providing a rich set of libraries for commonly needed tasks, as well as simple interface and logical structure to access these libraries.

CodeIgniter also uses Model-View-Controller (MVC) pattern in coding which separates the presentation and logic parts. Models are PHP classes that are designed to work with information in the database. View is the web page as the user sees it. Controller loads the view and manages any functions to be done in the database.

The system also uses AngularJS which allows in dynamic views of web applications. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit your unique development workflow and feature needs.

System Evolution

The system underwent different versions throughout its creation. The first request was a website dedicated to a test maker with the student information as a minor addition. The test maker was supposed to be the main focus of the project but due to some problems we focused on the student information page first.

The problem with the test maker was evaluation of the answers. The Guidance office uses different forms of questionnaires i.e. multiple choice with multiple answer values. A solution was proposed to create a table with the different answers having an initial value of zero. Every time an answer is selected it has different specific manipulations in the answer table.

Upon completion of the website excluding the test maker evaluation, a new problem arose. The Guidance office decided to remove the test maker function. This was due to a copyright infringement on some questionnaires that was supposed to be entered to the test maker page.

A substitution for this was the Risk Protective Assessment Survey which is made by the University of the Philippines – Baguio Campus Guidance office so no copyright infringement would be incurred.

The final version of the system included the student information page, the Risk Protective Assessment Survey, and the administrator accounts.

Appendices

The CodeIgniter framework used in the system needs a minimum of PHP version 5.6 but anything newer is recommended. The system also supports the following databases:

* MySQL version 5.1 and above through mysql, mysqli, and pdo drivers
* Oracle through the oci8 and pdo drivers
* PostogreSQL through the postogre and pdo drivers
* MS SQL through the mssql, sqlsrv, and pdo drivers
* SQLite through the sqlite, sqlite3, and pdo drivers
* CUBRID through the cubrid and pdo drivers
* Interbase/Firebird through the ibase and pdo drivers
* ODBC through the odbc and pdo drivers.

Index