Shubhrika Seghal, TJ Medi, Kinori Rosnow

INFSCI 2710 Database Management Systems

December 3, 2019

Project Report

1. Introduction

Attawheed Islamic Center is a community center that runs a weekend school for children in the Islamic community. The community center generally runs 35-40 weekends per year and currently serves over 150 students. As their school increases in scale the administration and operation of the school also becomes more involved.

The administrators currently use excel spreadsheets to keep track of operational data. Updating excel spreadsheets every time there is a change wastes resources to maintain consistency across all of the spreadsheets. They would like an information application that is consistent across the entire system. We built a relational database system that users interact with through a web application.

1. Purpose

We aim to create a central information system for Attawheed Islamic Center members. Access will be based on each member’s role in a given program and in the weekend school. Members can be a teacher of a class in a program, attend classes as a student or be an administrator with more information system privileges.

1. Intended Users (Members of the community center)
   1. Parents and guardians of the students – Most students are young and will probably not use the system themselves. Any that do, can access it through their family account which will be owned by a parent(s). Guardians or “Contacts” are people who are allowed to pick up students that may or may not be members of the community center.
   2. Teachers – Community members can teach classes to be classified as a teacher for that class. They can themselves also be students in another program.
   3. Administrators – Administrators are community members with privileges that allow them the authority to run parts of the school. They can make operational decisions in the information system such as change student enrollment or check for tuition payment status.
2. Intended Use
   1. Member Information: The system will hold information about members and which programs they are affiliated with and what the nature of their affiliation is (ie. student or teacher). An example is tracking the progress of the students like grades, classes, attendance, fees, etc. within a program.
   2. School Administrative Information: Administrators will be have the most privilege in the system to maintain organization. This role will allow them to check for proper inputs from other members as well as edit many of the fields that are often susceptible to exploitation attempts such as paid hours. Other examples of controlled relations are the expenses from a program and program enrollment.
   3. General Information Query: The database has the potential to be queried for information about their members. An example of this is finding out who has not yet paid off their tuition balance.
3. Scope
   1. Create a database system that holds information that is accessible and editable where applicable by a web application.
   2. The system will allow for tracking student progress in programs, teacher work hours and other relevant information.
4. Assumptions and Dependencies : Instead of assumptions we have user requirements
   1. Users will all be stored in the same table. Accounts that access the member portal can have multiple members (family members) assigned to them. Each member only has one family. Members can be teachers, students or administrators depending on the program.
   2. The timesheet will store the hours worked by each teacher with a maximum limit set by the administrator. The teacher’s volunteer and work hours will be paid accordingly. Administrators will be able to update fields through their portal to prevent teachers from over logging hours.
   3. Since keeping enrollment consistent in the system can prove to be difficult from the administrator’s perspective, students cannot be removed from the system after they are enrolled in a program. In order to remove a student, an administrator must remove the student from all programs first.
   4. Emergency contact information is required for kids and will default to the first parent stored in the system.
   5. Teacher Session   
      Privacy Thing
5. Database – describe relations here
   1. Database Tables:
   * Users – This table stores information about individual community center members. These members can be parents of the students enrolled in the programs offered, the students themselves or the teachers helping with the classes. All users will each have 1 family account they are linked to.
   * Family – Users will be associated with a family account. Each family will have 1 set of credentials to access the users’ information.
   * Contact – This table contains information about the people who are allowed to pick up or drop off the students from school.
   * Payment – This table keeps track of the payments made by the Family Account Holders for the members of that family. Since partial payments is allowed we are also calculation ‘balance due’ in this table.
   * Program – Members can be enrolled in 1 or more programs offered by the school. Each program can consist of 1 or more classes.
   * Enroll – This is a many to many relationship set that joins the users table with the program table.
   * Classes – This table stores information about the classes being offered at the school. Each class is a part of a program hence program\_id is added as an attribute to link the tables.
   * Assignment – This table has general details about the assignments that are given to the students enrolled in classes.
   * Student – This is relationship set between users and the classes they are enrolled in. It also stores grades that the students in received in the assignments of their respective classes.
   * Attendance – This table stores the attendance information of the students. Attendance is recorded for the day and not for each class the students attend.
   * Teacher\_info – this table has a “IS A” relationship with the users table. It stores information about the members of the school who are teaching 1 or more classes. Teachers can have an hourly pay or be volunteers.
   * Teaches – teachers teach classes. This is a many to many relationship requiring a separate table of primary keys from both tables (IDs).
   * Time\_sheet – This table keeps track of the hours worked by the staff or the teachers in the school. Also calculates the amount that is to paid to the teachers and when they were last paid.
   * Expense Sheet – Programs may have expenses reimbursed by the school that will be logged in the system. This table stores the merchant and item information and also how payment was made for a particular expense.
   1. Normal Forms:

All tables do not have any transitive dependencies on the candidate key meaning they are all 3rd Normal Form. In addition there are no primary key attributes that are dependent on an attribute not part of the candidate key. Therefore all tables are also BCNF.

Example Decomposition:

* Original Table:

R (assignments): user\_id, class\_id, assignment\_id, grade, total\_points, due\_date, description

FD 1: user\_id, class\_id, assignment\_id 🡪 grade

FD 2: assignment\_id 🡪 total\_points, due\_date, description

FD 3: assignment\_id 🡪 class\_id ???

* Good Split:

R1 (graded): user\_id, class\_id, assignment\_id, grade

R2(assignment): assignment\_id, total\_points, due\_date, description

* Suboptimal Split: This decomposition is lossy in that it loses the information by not keeping assignment\_id associated with the grade.

R1: user\_id, class\_id 🡪 grade

R2: class\_id, assignment\_id 🡪 total\_points, due\_date, description

1. User Interface/Web Application

The user interface will start the user with a login screen. The user role defines the privileges they have. Based on these privileges their view will also differ. There will be three types of access: parents, teachers and administrators. All screens will have the abilities of a user, but privileged accounts will also have tabs for their respective job roles.

* 1. User screen – child information, option of adding guardians, track of student enrollment/performance (classes, grades and attendance), a screen to track tuition and fees.
  2. Teacher screen – grade/attendance input, classes, parents/students information, timesheet, program expense sheet.
  3. Admins screen – full access to everything, update, insert and delete all the fields (that can be updated). In addition they will have access to students’ tuition status, teacher class assignments and teacher payment info.

1. Front/Back Connection Description

The back end is a database of tables each with a separate controller. The controllers interact with the tables to query or update the table. The tables were created with MySQL DDL then migrated in with migration generators. The controller acts as an intermediary between a model and the controller’s respective table. The controller is created by php artisan. The models are then combined to create the website which can be viewed and interacted with using their UI. Finish describing technologies used here.

1. Security

In addition to the system needs, the user will be storing student information. This data is FERPA regulated and requires securing from public access. By another user’s suggestion we have decided to use the Laravel framework to build out the system. The community center wants to host the web app and data in an Azure cloud solution. We have advised they opt for the security required to store their data. It would still be best for a security professional to review the system before putting student data in the system.

1. Testing Description
   1. Database Creation
   2. Database UI Interactions
2. Limitations/Future Additions/Potential Improvements:
   1. Mobile application – The school would like to eventually use the web application to then be linked to a mobile application for convenience.
   2. Security – The Laravel Framework handles some of the security, but this should be reviewed to potential improvements.
   3. Verification System?
   4. History logs – There currently isn’t a history log system. A history log system would allow for an undo button in the event of a user error.
   5. Metadata Tables – Metadata tables would allow

Components- what do they know, major components, DB specifications, hosting

Interfaces between components – APIs, potential problems

**SQL Table Creation Statements:**

CREATE SCHEMA `school`;

CREATE TABLE `school`.`family` (

`family\_id` INT NOT NULL,

`address` VARCHAR(50) NOT NULL,

PRIMARY KEY (`family\_id`)

);

CREATE TABLE `school`.`users` (

`users\_id` INT NOT NULL,

`family\_id` INT NOT NULL,

`name` VARCHAR(50) NOT NULL,

`gender` VARCHAR(10),

`dob` DATE NOT NULL,

`age` INT,

`email` VARCHAR(30) NOT NULL,

`secondary\_email` VARCHAR(30),

`primary\_phone` BIGINT NOT NULL,

`secondary\_phone` BIGINT NOT NULL,

`city` VARCHAR(20) NOT NULL,

`state` VARCHAR(20) NOT NULL,

`zip\_code` INT NOT NULL,

`is\_parent` VARCHAR(10) NOT NULL,

`password` VARCHAR(20) NOT NULL,

`allergies` VARCHAR(50),

`is\_photo` BOOLEAN NOT NULL,

`emergency\_name` VARCHAR(50),

`emergency\_contact` BIGINT,

####`emergency\_relation` VARCHAR(15),

PRIMARY KEY (`users\_id`),

FOREIGN KEY (`family\_id`) REFERENCES family(`family\_id`)

);

CREATE TABLE `school`.`contact` (

`name` VARCHAR(20) NOT NULL,

`contact\_id` INT NOT NULL,

`phone\_number` BIGINT NOT NULL,

`users\_id` INT NOT NULL,

PRIMARY KEY (`contact\_id`),

FOREIGN KEY (`users\_id`) REFERENCES users (`users\_id`)

);

CREATE TABLE `school`.`payment` (

`payment\_id` INT NOT NULL,

`users\_id` INT NOT NULL,

`paid\_on` DATE NOT NULL,

`payment\_method` VARCHAR(20) NOT NULL,

`status` VARCHAR(10) NOT NULL,

`total\_fees` FLOAT NOT NULL,

`fees\_paid` FLOAT NOT NULL,

`balance\_due` FLOAT NOT NULL,

PRIMARY KEY (`payment\_id`),

FOREIGN KEY (`users\_id`) REFERENCES users(`users\_id`)

);

CREATE TABLE `school`.`program` (

`program\_id` INT NOT NULL,

`description` VARCHAR(100) NOT NULL,

`start\_date` DATE NOT NULL,

`end\_date` DATE NOT NULL,

`fees` FLOAT NOT NULL,

`age\_ll` INT NOT NULL,

`age\_up` INT NOT NULL,

PRIMARY KEY (`program\_id`)

);

CREATE TABLE `school`.`enroll` (

`users\_id` INT NOT NULL,

`program\_id` INT NOT NULL,

PRIMARY KEY (`users\_id`,`program\_id`),

INDEX (`users\_id`),

INDEX(`program\_id`),

FOREIGN KEY (`users\_id`) REFERENCES users(`users\_id`),

FOREIGN KEY (`program\_id`) REFERENCES program(`program\_id`)

);

CREATE TABLE `school`.`classes` (

`classes\_id` INT NOT NULL,

`program\_id` INT NOT NULL,

`name` VARCHAR(50) NOT NULL,

`schedule` VARCHAR(50) NOT NULL DEFAULT 'TBD',

`fees` FLOAT NOT NULL,

PRIMARY KEY (`classes\_id`),

FOREIGN KEY (`program\_id`) REFERENCES program(`program\_id`)

);

CREATE TABLE `school`.`assignment` (

`assignment\_no` VARCHAR(10) NOT NULL,

`classes\_id` INT NOT NULL,

`total\_points` INT NOT NULL,

`description` VARCHAR(50) NOT NULL,

`due\_date` DATE NOT NULL,

PRIMARY KEY (`assignment\_no`),

FOREIGN KEY (`classes\_id`) REFERENCES classes(`classes\_id`)

);

CREATE TABLE `school`.`student` (

`users\_id` INT NOT NULL,

`classes\_id` INT NOT NULL,

`assignment\_no` VARCHAR(10) NOT NULL,

`grade` VARCHAR(3) NOT NULL DEFAULT 'TBD',

PRIMARY KEY (`users\_id`,`classes\_id`,`assignment\_no`),

INDEX (`users\_id`),

INDEX(`classes\_id`),

INDEX (`assignment\_no`),

FOREIGN KEY (`users\_id`) REFERENCES users(`users\_id`),

FOREIGN KEY (`classes\_id`) REFERENCES classes(`classes\_id`),

FOREIGN KEY (`assignment\_no`) REFERENCES assignment(`assignment\_no`)

);

CREATE TABLE `school`.`attendance` (

`users\_id` INT NOT NULL,

`dates` DATE NOT NULL,

`isattended` BOOLEAN NOT NULL,

PRIMARY KEY (`users\_id`,`dates`),

FOREIGN KEY (`users\_id`) REFERENCES users(`users\_id`)

);

CREATE TABLE `school`.`teacher\_info` (

`users\_id` INT NOT NULL,

`start\_date` DATE NOT NULL,

`volunteer` VARCHAR(10) NOT NULL,

`role` VARCHAR(20) NOT NULL,

`status` VARCHAR(10),

PRIMARY KEY (`users\_id`),

FOREIGN KEY (`users\_id`) REFERENCES users(`users\_id`)

);

CREATE TABLE `school`.`teaches` (

`users\_id` INT NOT NULL,

`classes\_id` INT NOT NULL,

PRIMARY KEY (`users\_id`,`classes\_id`),

INDEX (`users\_id`),

INDEX(`classes\_id`),

FOREIGN KEY (`users\_id`) REFERENCES teacher\_info(`users\_id`),

FOREIGN KEY (`classes\_id`) REFERENCES classes(`classes\_id`)

);

CREATE TABLE `school`.`time\_sheet` (

`users\_id` INT NOT NULL,

`hourly\_rate` FLOAT NOT NULL,

`dates` DATE NOT NULL,

`total\_hours` INT NOT NULL DEFAULT 0,

`amount` FLOAT NOT NULL,

`paid\_on` DATE NOT NULL,

PRIMARY KEY (`users\_id`,`dates`,`hourly\_rate`),

FOREIGN KEY (`users\_id`) REFERENCES teacher\_info(`users\_id`)

);

CREATE TABLE `school`.`expense\_sheet` (

`expense\_id` INT NOT NULL,

`program\_id` INT NOT NULL,

`item` VARCHAR(20) NOT NULL,

`description` VARCHAR(100),

`merchant` VARCHAR(20) NOT NULL,

`expense` FLOAT NOT NULL,

`payment\_type` VARCHAR(10) NOT NULL,

`status` VARCHAR(10) NOT NULL,

`dates` DATE NOT NULL,

PRIMARY KEY (`expense\_id`),

FOREIGN KEY (`program\_id`) REFERENCES program(`program\_id`)

);