

**RISE Learning Institute Database**

**Computer Scientists:**

Instructor: Dr. Sara Farag

Marius Popescu

Anara Satkeeva

Jeffrey Talada

Erik Uri

## I. Project proposal Overview

Our group, “group 3”, consisting of Marius Popescu, Anara Satkeeva, Jeffrey Talada and Erik Uri, proposes creating a new database for the RISE Learning Institute. The purpose of our project is to create a web interface and database to browse, add, store and update the data that advisors are collecting in their meetings with students to better aid students in their career efforts.

## The Problem

* The Rise Learning Institute and the Center of Careers Connections at Bellevue College work to connect students, local businesses, and educators together to create a myriad of services for the school, students, and the community, such as learning service opportunities, internships, jobs, advisory boards and grants to further enrich the learning environment here on campus.
* Different staff members at RISE collect and store these contacts and relationships in several different ways, but there isn’t a standard used by everyone.
* Information gets lost in the cracks when different staff members at RISE pick up where other members have left off, restricting the institute from accomplishing more.
* RISE also wants to be able to query and report on the demographics of students that they work with based off of ZIP codes, for the purpose of getting grants.

**Client Requirements**

* Keep track of students/community members and their demographic information
* Must be able to generate reports by zip code with demographic filters
* Keep track of advising sessions, limit three for non-students.
* A student may meet with different advisors
* Advisors want to see job/internship history and any past notes
* Time stamp student internship and job application
  + Keep track of successful applications when possible
  + Be able to retroactively add a history of internships or job applications
* Each relationship with an organization may have a different contact
* Relationships with organizations may have a RISE lead who maintains the relationship
* Wants to be able to report number of relationships of which types with all organizations

## Our Solution

* We will create a web interface and database to create, read, update, and delete these relationships to allow for easy and self-explanatory use.
* We will also provide a way to import/export from/to (CSV), so RISE can bulk collect their data in excel or upgrade to a CRM later on.
* We will provide a way for the RISE staff to generate and export reports.

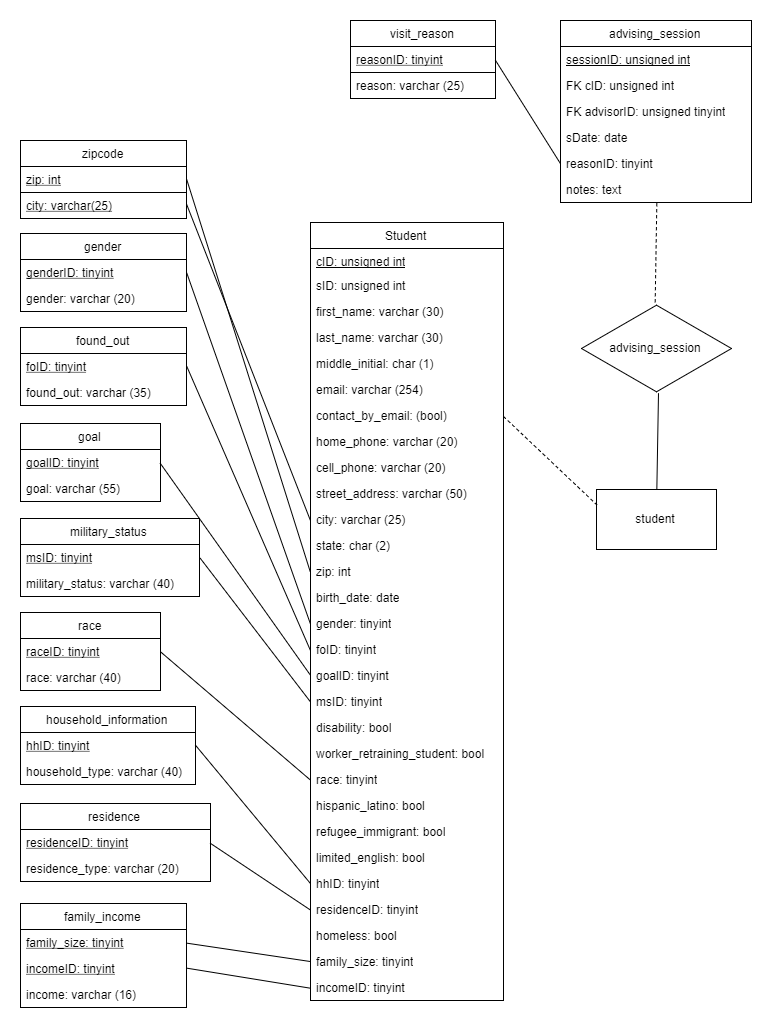
## II. Database Design

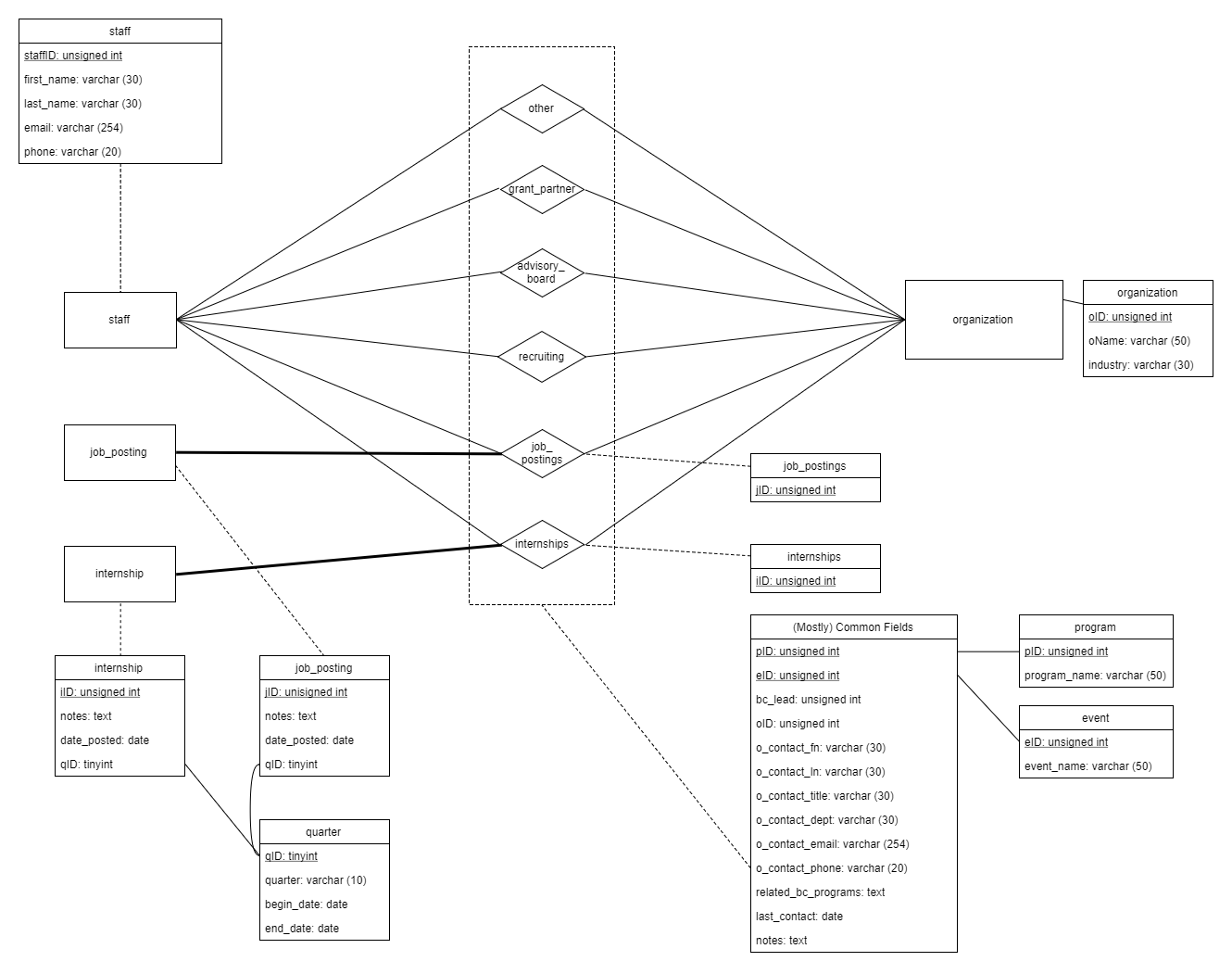
The following diagram provide a high-level introduction to our database. It is provided as reference while learning about the system’s client functionality in the following section.

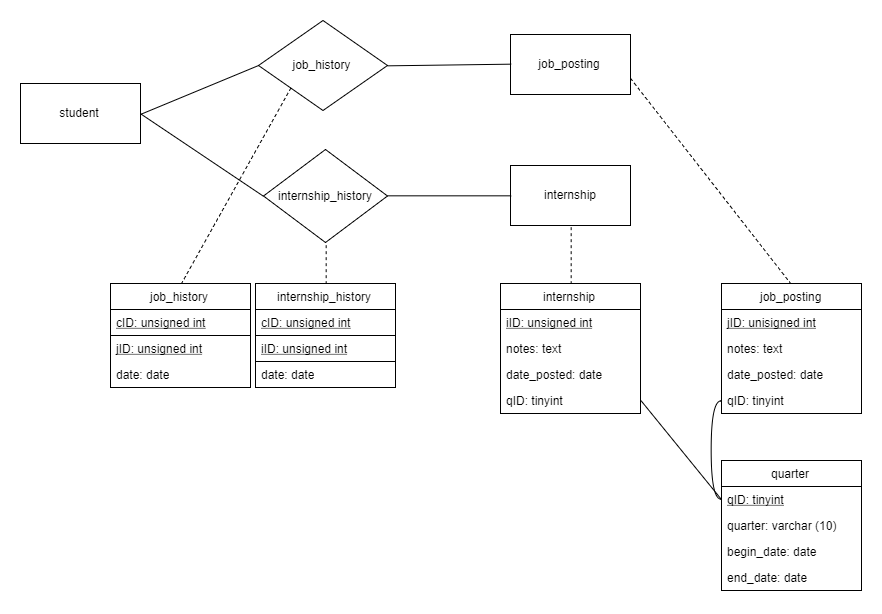


Dashed lines represent expansions to show all the attributes of that entity.

Solid lines represent actual connections.



****

****

**Tables**

This section describes what data we will store in our database as well as how our relations represent this data. The following tables will be included in our database. Please refer to the E-R diagrams at the beginning of the document for a higher-level graphical description.

CREATE TABLE zipcode (

zip INT,

city VARCHAR(25),

PRIMARY KEY (zip , city)

);

This table stores the zip code and the city.

CREATE TABLE found\_out (

foID TINYINT AUTO\_INCREMENT PRIMARY KEY,

found\_out VARCHAR(35)

);

This table will have two columns one for found out ID (a number), and one for the found out name exp.: web, other.

CREATE TABLE goal (

goalID TINYINT AUTO\_INCREMENT PRIMARY KEY,

goal VARCHAR(55)

);

This table will have two columns one for goal ID (a number), and one for the goal name exp.: transfer degree.

CREATE TABLE military\_status (

msID TINYINT AUTO\_INCREMENT PRIMARY KEY,

military\_status VARCHAR(40)

);

This table will have two columns one for military status ID (a number), and one for the military status name exp.: US Military.

CREATE TABLE race (

raceID TINYINT AUTO\_INCREMENT PRIMARY KEY,

race VARCHAR(40)

);

This table will have two columns one for race ID (a number), and one for the race name exp.: White/Caucasian.

CREATE TABLE household\_information (

hhID TINYINT AUTO\_INCREMENT PRIMARY KEY,

household\_type VARCHAR(40)

);

This table will have two columns one for household information ID (a number), and one for the household information name exp.: Adult Male.

CREATE TABLE residence (

residenceID TINYINT AUTO\_INCREMENT PRIMARY KEY,

residence\_type VARCHAR(24)

);

This table will have two columns one for residence ID (a number), and one for the residence name exp.: 'Live outside city limits'

.

CREATE TABLE family\_income (

family\_size TINYINT,

incomeID TINYINT,

income VARCHAR(16),

PRIMARY KEY (family\_size , incomeID)

);

This table will have three columns one for income ID (a number), one for family size, and one for the income value.

CREATE TABLE student (

cID INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

sID INT UNSIGNED,

first\_name VARCHAR(30) NOT NULL,

last\_name VARCHAR(30) NOT NULL,

middle\_initial CHAR(1),

email VARCHAR(254),

contact\_by\_email BOOLEAN,

home\_phone VARCHAR(20),

cell\_phone VARCHAR(20),

street\_address VARCHAR(50),

city VARCHAR(25),

state CHAR(2),

zip INT,

birth\_date DATE,

genderID TINYINT,

foID TINYINT,

goalID TINYINT,

msID TINYINT,

disability BOOLEAN,

worker\_retraining\_student BOOLEAN,

race TINYINT,

hispanic\_latino BOOLEAN,

refugee\_immigrant BOOLEAN,

limited\_english BOOLEAN,

hhID TINYINT,

residenceID TINYINT,

homeless BOOLEAN,

family\_size TINYINT,

incomeID TINYINT,

FOREIGN KEY (zip , city)

REFERENCES zipcode (zip , city),

FOREIGN KEY (genderID)

REFERENCES gender (genderID),

FOREIGN KEY (foID)

REFERENCES found\_out (foID),

FOREIGN KEY (goalID)

REFERENCES goal (goalID),

FOREIGN KEY (msID)

REFERENCES military\_status (msID),

FOREIGN KEY (race)

REFERENCES race (raceID),

FOREIGN KEY (hhID)

REFERENCES household\_information (hhID),

FOREIGN KEY (residenceID)

REFERENCES residence (residenceID),

FOREIGN KEY (family\_size , incomeID)

REFERENCES family\_income (family\_size , incomeID)

);

All information about a student will be input in this table.

CREATE TABLE staff (

staffID INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(30) NOT NULL,

last\_name VARCHAR(30) NOT NULL,

email VARCHAR(254) NOT NULL,

phone VARCHAR(20) NOT NULL

);

This table will have columns for first name, last name, email and phone for the staf members.

CREATE TABLE visit\_reason (

reasonID TINYINT AUTO\_INCREMENT PRIMARY KEY,

reason VARCHAR(19)

);

The purpose of this table is to store the visit reason. It will have two columns one for reason ID (a number), and one for the reason name exp.: 'Career Planning'

CREATE TABLE advising\_session (

sessionID INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

cID INT UNSIGNED NOT NULL,

adviserID INT UNSIGNED NOT NULL,

sDate DATE,

reasonID TINYINT,

notes TEXT,

FOREIGN KEY (cID)

REFERENCES student (cID),

FOREIGN KEY (adviserID)

REFERENCES staff (staffID),

FOREIGN KEY (reasonID)

REFERENCES visit\_reason (reasonID)

);

This table will have the student cID, adviser ID and reason ID like foreign keys and other columns that will be completed in the time of the advising session.

CREATE TABLE quarter (

qID TINYINT AUTO\_INCREMENT PRIMARY KEY,

quarter VARCHAR(10),

begin\_date DATE,

end\_date DATE

);

This table will keep information about quarter, the beginning, the end, the quarter Id and the quarter name.

CREATE TABLE workshop (

wID TINYINT AUTO\_INCREMENT PRIMARY KEY,

wName VARCHAR(30)

);

This table will have two columns one for workshop ID (a number), and one for the workshop name

CREATE TABLE job\_posting (

jID INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

date\_posted DATE,

qID TINYINT,

notes TEXT,

FOREIGN KEY (qID)

REFERENCES quarter (qID)

);

This table will keep information about active job postings.

CREATE TABLE job\_history (

cID INT UNSIGNED NOT NULL,

jID INT UNSIGNED NOT NULL,

date\_hired DATE,

PRIMARY KEY (cID , jID),

FOREIGN KEY (cID)

REFERENCES student (cID),

FOREIGN KEY (jID)

REFERENCES job\_posting (jID)

);

This table will keep a history about job postings and hiring’s

CREATE TABLE internship\_history (

cID INT UNSIGNED NOT NULL,

iID INT UNSIGNED NOT NULL,

date\_hired DATE,

PRIMARY KEY (cID , iID),

FOREIGN KEY (cID)

REFERENCES student (cID),

FOREIGN KEY (iID)

REFERENCES internship (iID)

);

This table will keep a history about internships postings and hiring’s

CREATE TABLE organization (

oID INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

oName VARCHAR(50),

industry VARCHAR(30)

);

This table will keep information about organizations that posting jobs and internships.

CREATE TABLE job\_postings (

jID INT UNSIGNED,

bc\_lead INT UNSIGNED,

oID INT UNSIGNED,

o\_contact\_fn VARCHAR(30),

o\_contact\_ln VARCHAR(30),

o\_contact\_title VARCHAR(30),

o\_contact\_dept VARCHAR(30),

o\_contact\_email VARCHAR(254),

o\_contact\_phone VARCHAR(20),

related\_bc\_programs TEXT,

notes TEXT,

PRIMARY KEY (jID , oID , bc\_lead),

FOREIGN KEY (jID)

REFERENCES job\_posting (jID),

FOREIGN KEY (oID)

REFERENCES organization (oID),

FOREIGN KEY (bc\_lead)

REFERENCES staff (staffID)

);

This table will keep a history about job postings

CREATE TABLE internships (

iID INT UNSIGNED,

bc\_lead INT UNSIGNED,

oID INT UNSIGNED,

o\_contact\_fn VARCHAR(30),

o\_contact\_ln VARCHAR(30),

o\_contact\_title VARCHAR(30),

o\_contact\_dept VARCHAR(30),

o\_contact\_email VARCHAR(254),

o\_contact\_phone VARCHAR(20),

related\_bc\_programs TEXT,

notes TEXT,

PRIMARY KEY (iID , oID , bc\_lead),

FOREIGN KEY (iID)

REFERENCES internship (iID),

FOREIGN KEY (oID)

REFERENCES organization (oID),

FOREIGN KEY (bc\_lead)

REFERENCES staff (staffID)

);

This table will keep a history about internship postings

CREATE TABLE program (

pID INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

program\_name VARCHAR(50)

);

This table will have two columns one for program ID (a number), and one for the program name

CREATE TABLE event (

eID INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

event\_name VARCHAR(50)

);

This table will have two columns one for event ID (a number), and one for the event name

CREATE TABLE recruiting (

pID INT UNSIGNED,

eID INT UNSIGNED,

bc\_lead INT UNSIGNED,

oID INT UNSIGNED,

o\_contact\_fn VARCHAR(30),

o\_contact\_ln VARCHAR(30),

o\_contact\_title VARCHAR(30),

o\_contact\_dept VARCHAR(30),

o\_contact\_email VARCHAR(254),

o\_contact\_phone VARCHAR(20),

last\_contact DATE,

related\_bc\_programs TEXT,

notes TEXT,

PRIMARY KEY (pID , eID , oID),

FOREIGN KEY (oID)

REFERENCES organization (oID),

FOREIGN KEY (bc\_lead)

REFERENCES staff (staffID),

FOREIGN KEY (pID)

REFERENCES program (pID),

FOREIGN KEY (eID)

REFERENCES event (eID)

);

This table keeps track of recruiters of various organizations and any recruiting events they attend

CREATE TABLE advisory\_board (

bc\_lead INT UNSIGNED,

oID INT UNSIGNED,

o\_contact\_fn VARCHAR(30),

o\_contact\_ln VARCHAR(30),

o\_contact\_title VARCHAR(30),

o\_contact\_dept VARCHAR(30),

o\_contact\_email VARCHAR(254),

o\_contact\_phone VARCHAR(20),

last\_contact DATE,

notes TEXT,

PRIMARY KEY (oID , bc\_lead),

FOREIGN KEY (oID)

REFERENCES organization (oID),

FOREIGN KEY (bc\_lead)

REFERENCES staff (staffID)

);

This table will keep information about advisory board members. It will include column like name, mail, phone, title and department.

CREATE TABLE grant\_partner (

bc\_lead INT UNSIGNED,

oID INT UNSIGNED,

o\_contact\_fn VARCHAR(30),

o\_contact\_ln VARCHAR(30),

o\_contact\_title VARCHAR(30),

o\_contact\_dept VARCHAR(30),

o\_contact\_email VARCHAR(254),

o\_contact\_phone VARCHAR(20),

last\_contact DATE,

related\_bc\_programs TEXT,

notes TEXT,

PRIMARY KEY (oID , bc\_lead),

FOREIGN KEY (oID)

REFERENCES organization (oID),

FOREIGN KEY (bc\_lead)

REFERENCES staff (staffID)

);

**Triggers**

DELIMITER $$

CREATE TRIGGER before\_interships\_update

BEFORE UPDATE ON internships

FOR EACH ROW

BEGIN

INSERT INTO intership

SET action = 'update',

oID = OLD.oID,

change date\_posted = NOW();

END$$

This trigger will automatically set the date from the internship table to the actual value when somebody to insert the data in the internships table.

DELIMITER $$

CREATE TRIGGER before\_job\_postings\_update

BEFORE UPDATE ON job\_postings

FOR EACH ROW

BEGIN

INSERT INTO job\_posting

SET action = 'update',

oID = OLD.oID,

change date\_posted = NOW();

END$$

This trigger will set automatically the date from the job posting table to the actual value when somebody to insert the data in the job postings table.

DELIMITER $$

CREATE TRIGGER before\_advisory\_board\_update

BEFORE UPDATE ON advisory\_board

FOR EACH ROW

BEGIN

INSERT INTO advising\_session

SET action = 'update',

cID = OLD.cID,

change date = NOW();

END$$

This trigger will set automatically the date from the advisory table to the actual value when somebody to insert the data in the advising session postings table.

**Views**

CREATE OR REPLACE VIEW Advisor

AS SELECT advising\_session.adviserID, advising\_session.cID, advising\_session.date, advising\_session.notes, advising\_session.reasonID, advising\_session.sessionID, job\_history.date\_hired, job\_history.jID, internship\_history.iID

FROM advising\_session, job\_history, internship\_history

WHERE advising\_session.cID = job\_history.cID = internship\_history.cID

WITH CHECK OPTION;

This is a view for an advisor. It display important fields for a advising session.

CREATE OR REPLACE VIEW StudentsByReason

AS SELECT student.first\_name, student.last\_name, advising\_session.notes

FROM student, advising\_session, visit\_reason

WHERE visit\_reason.reasonID = advising\_session.reasonID AND student.cID = advising\_session.cID

WITH CHECK OPTION;

This view will allow the user to visualize the students first and last name and the notes from advising session. The students will be sorted by visit reason.

CREATE OR REPLACE VIEW StudentsByDate

AS SELECT student.first\_name, student.last\_name

FROM student, advising\_session

WHERE advising\_session.date = '11.10.2017'

WITH CHECK OPTION;

This view will allow the user to visualize the students first and last name. The students will be sorted by visit date.

CREATE OR REPLACE VIEW AdminJobs

AS SELECT job\_postings.o\_contact\_title, job\_postings.related\_bc\_programs, recruiting.event\_name, recruiting.notes, recruiting.program\_name, recruiting.oID

FROM recruiting, job\_postings, internships

WHERE recruiting.oID = job\_postings.oID = internships.oID

WITH CHECK OPTION;

This view will allow the administrator to view more meaningful field from job postings.

CREATE OR REPLACE VIEW AdminAdvisoring

AS SELECT advising\_session.adviserID, advising\_session.date, student.first\_name, student.last\_name

FROM advising\_session, student

WHERE advising\_session.cID = student.cID

WITH CHECK OPTION;

This view will allow the administrator to view information’s from advising sessions.

CREATE OR REPLACE VIEW Interships

AS SELECT \*

FROM internships

WITH CHECK OPTION;

This view will display the internships.

CREATE OR REPLACE VIEW JobPostings

AS SELECT \*

FROM job\_postings

WITH CHECK OPTION;

This view will display the job postings.

CREATE OR REPLACE VIEW Organizations

AS SELECT \*

FROM organization

WITH CHECK OPTION;

This view will display the organizations.

**III. Client Functionalities**

This section describes how our user interface will work and how RISE staff will view and be able to interact with the web site and database.

III.A. The home page

The home page consists of six buttons leading to different sets of functionality. There is also a navigation bar that is common to every page and links to the most commonly used functions.

III.B. Advising

The advising page deals with managing an advising session. Type in a name or student ID and it will pull up the person’s history if they have one and relevant data. An advisor will be able to see past notes as well as make new notes.

III.C. Job search

Job search is fairly simple. You can choose one of several fields to search by including for example, industry, and it will list available job postings.

III.D. Internship search

This is similar to job search, allows one to search for internships based on a chosen field.

III.E. Administration

This page allows one to insert and modify data.

III.F. Reporting

This page generates reports based on the selected criteria. Select the report you want to generate and any filters you’d like to apply and the database will create a .csv file.

III.G. Import/Export

This allows one to import and export data to and from .csv files. These files are compatible with excel.

If anything should happen to the database itself there are several steps to resetting the database.

III.H. Drop Tables

There is a file called drop\_tables. Running this file in command prompt while connect to the database will delete all the tables.

III.I. Create Tables

Next there is a file called create\_tables that will rebuild all the tables listed in section II. It will also insert form selection options. Any options that have been added after the original install will have to be reinserted manually.

To insert a new option manually follow the following format and run in command prompt while connected to the database:

INSERT INTO table\_name (field, field, field) values

(data, data, data),

(data, data, data);

Don’t include any self-incrementing fields when inserting. Which fields are self-incrementing can be found in section II.

III.J. The database is now rebuilt and you may import your data.

**IV. Installation Guide for the system**

This document has been designed to guide you through the installation of the database.

*Step I: Setup the Web Server*

Download and install WampServer. Follow the setup instructions.

*Step II: Create the Database*

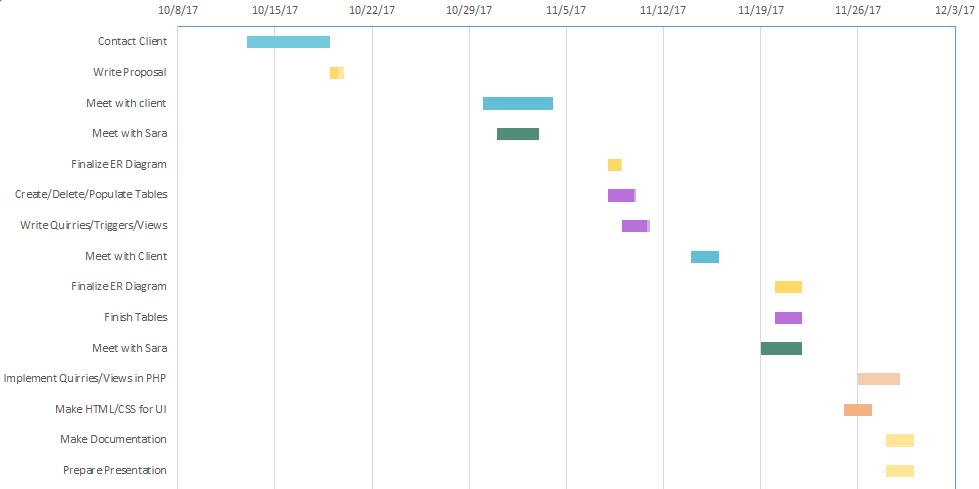
From the database files place the folder htdocs/php into c:\wamp\www assuming you used the defaults when installing WampServer. Place the rest of the database files into your preferred directory. Click on the WampServer icon and in server settings set the directory of the files.

Open the MySQL Console using the Wamp icon and copy/paste the create tables code into the console and press enter.

*Step III: Test the Site*

In your browser, go to “localhost” and now you can add your data using the site.

**V. Project timeline – Gantt Chart**

****

* **Week 1 + 2**
  + **Fumbling for a client and setting up a first meeting**
* **Week 3**
  + **Maybe we'll get feedback to help with the next client meeting?**
* **Week 4**
  + **Schedule with client, oh wait feedback**
* **Week 5**
  + **Can't physically meet together, start using Github and rush to finish milestone 2**
* **Week 6**
  + **Client is ecstatic with design we've presented to him**
* **Week 7 + 8**
  + **Erik requests we present first at meeting with Sara**
  + **UI critical team member goes into food coma during crunch time**

**VI. Group Evaluation**

# Group Self Evaluation Checklist

Name Jeff, Erik, Ana, Marius Class Period Fall 2017 Date 11/30/17

Topic of Study CS331 Group Number Group 3

As a team, decide which answer best suits the way your team worked together. Then, complete the remaining sentences.



We finished our task on time, and we did a good job!

YES

NO



We encouraged each other and we cooperated with each other.

YES

NO



We used quiet voices in our communications.

YES

NO



We each shared our ideas, then listened and valued each other’s ideas.

YES

NO

We did best at meetings with the client.

Next time we could improve at balancing work load and working together.

**VII. Client Approval**

CLIENT NAME: Michael Reese (RISE Learning Institute Director)

CLIENT SIGNATURE:

COMPUTER SCIENTISTS:

Instructor: Dr. Sara Farag

Marius Popescu

Anara Satkeeva

Jeffrey Talada

Erik Uri

DATE: 11/30/2017