**DATABASE ADMINISTRATION DOCUMENTATION**

**TABLE OF CONTENTS**

CREATE TABLE STATEMENTS 2

DESCRIPTIONS OF THE FILES 68

LOG-IN AUTHENTICATION SYSTEM 69

SECURITY 72

FUNCTIONS 72

PHP-EXECUTION FILES 86

IMPORTING 94

**CREATE TABLE Statements:**

CREATE DATABASE rebus;

USE rebus;

# Create Tables

# Table to hold all organizations

CREATE TABLE IF NOT EXISTS organization

(DROP DATABASE rebus;

org\_id INT NOT NULL AUTO\_INCREMENT,

org\_name VARCHAR(75) NOT NULL,

org\_address VARCHAR(100) NOT NULL,

org\_city VARCHAR(60) NOT NULL,

org\_state\_us CHAR(2),

org\_state\_other CHAR(5),

org\_zip VARCHAR(15) NOT NULL,

org\_country CHAR(5),

org\_parent\_branch INT,

org\_mnged\_serv\_provider ENUM('Y','N'),

org\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

org\_add\_uname VARCHAR(100),

org\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

org\_update\_uname VARCHAR(100),

PRIMARY KEY (org\_id),

FOREIGN KEY (org\_parent\_branch) REFERENCES organization (org\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

/\* Index to help speed up searches and make org\_name UNIQUE \*/

CREATE UNIQUE INDEX organization\_uk ON organization (org\_name);

CREATE TABLE IF NOT EXISTS user

(

user\_id INT NOT NULL AUTO\_INCREMENT,

user\_fname VARCHAR(60) NOT NULL,

user\_midname VARCHAR(60),

user\_lname VARCHAR(60) NOT NULL,

user\_country\_code CHAR(3),

user\_phone\_num VARCHAR(16) NOT NULL,

user\_email VARCHAR(100) NOT NULL, # this field servers as the person's username

/\* Passwords are hashed during user creation or password resets from website \*/

user\_password VARCHAR(255) NOT NULL,

user\_temp\_flag ENUM('Y', 'N') NOT NULL DEFAULT 'N',

user\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

user\_add\_uname VARCHAR(100),

user\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

user\_update\_uname VARCHAR(100),

org\_id INT,

user\_manager INT,

PRIMARY KEY (user\_id),

FOREIGN KEY (org\_id) REFERENCES organization (org\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (user\_manager) REFERENCES user (user\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Speeds to retrieval for logins and determining session information and makes user\_email UNIQUE

CREATE UNIQUE INDEX user\_uk ON user (user\_email);

# Table to hold all the different flavors of a system, should be a small table

CREATE TABLE IF NOT EXISTS sys\_type

(

type\_id INT NOT NULL AUTO\_INCREMENT,

type\_name VARCHAR(75) NOT NULL,

type\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

type\_add\_uname VARCHAR(100),

type\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

type\_update\_uname VARCHAR(100),

PRIMARY KEY (type\_id)

) ENGINE=InnoDB;

# Help speed retrieval as type\_name is used in certain queries and makes system type unique

CREATE UNIQUE INDEX sys\_type\_uk ON sys\_type (type\_name);

# Holds all information for each system an organization keeps scan results for

CREATE TABLE IF NOT EXISTS system

(

sys\_id INT NOT NULL AUTO\_INCREMENT,

sys\_name VARCHAR(100) NOT NULL,

sys\_ip\_address VARCHAR(20),

sys\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

sys\_add\_uname VARCHAR(100),

sys\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

sys\_update\_uname VARCHAR(100),

type\_id INT NOT NULL,

org\_id INT NOT NULL,

PRIMARY KEY (sys\_id),

FOREIGN KEY (type\_id) REFERENCES sys\_type (type\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (org\_id) REFERENCES organization (org\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Help speed up queries where a list of systems for each organization is needed and makes system names unique within a given organization

CREATE UNIQUE INDEX system\_uk ON system (org\_id, sys\_name);

# This table will be used to hold organization's reasonings for complying or not complying with a requirement

CREATE TABLE IF NOT EXISTS artifact

(

art\_id INT NOT NULL AUTO\_INCREMENT,

art\_text VARCHAR(255) NOT NULL,

art\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

art\_add\_uname VARCHAR(100),

art\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

art\_update\_uname VARCHAR(100),

org\_id INT NOT NULL,

PRIMARY KEY (art\_id),

FOREIGN KEY (org\_id) REFERENCES organization (org\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to speed up retrieval when entering results manually and set artifact text unique to the organization

CREATE UNIQUE INDEX artifact\_uk ON artifact (org\_id, art\_text);

# Table to hold all standards needed

CREATE TABLE IF NOT EXISTS standard

(

stand\_id INT NOT NULL AUTO\_INCREMENT,

stand\_name VARCHAR(100) NOT NULL,

stand\_version\_rev\_num VARCHAR(100) NOT NULL,

stand\_effective\_date DATE NOT NULL,

stand\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

stand\_add\_uname VARCHAR(100),

stand\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

stand\_update\_uname VARCHAR(100),

stand\_root INT,

PRIMARY KEY (stand\_id),

FOREIGN KEY (stand\_root) REFERENCES standard (stand\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to speed up queries where a standard name and revision number are needed and the pair is unique

CREATE UNIQUE INDEX standard\_uk ON standard (stand\_name, stand\_version\_rev\_num);

# Holds the categories for each standard, some categories can be used by more than one standard

CREATE TABLE IF NOT EXISTS category

(

cat\_id INT NOT NULL AUTO\_INCREMENT,

cat\_name VARCHAR(75) NOT NULL,

cat\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

cat\_add\_uname VARCHAR(100),

cat\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

cat\_update\_uname VARCHAR(100),

PRIMARY KEY (cat\_id)

) ENGINE=InnoDB;

# Needed to speed up retrieval of category names and numbers when needed and make the pair unique

CREATE UNIQUE INDEX category\_uk ON category (cat\_name);

# Bridge entity to account for the use of categories by multiple standards and their revisions

CREATE TABLE IF NOT EXISTS standard\_category

(

standcat\_id INT NOT NULL AUTO\_INCREMENT,

stand\_id INT NOT NULL,

cat\_id INT NOT NULL,

standcat\_num VARCHAR(10) NOT NULL,

standcat\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

standcat\_add\_uname VARCHAR(100),

standcat\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

standcat\_update\_uname VARCHAR(100),

PRIMARY KEY (standcat\_id),

FOREIGN KEY (stand\_id) REFERENCES standard (stand\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (cat\_id) REFERENCES category (cat\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to make sure each standard and category combination only occurs once

CREATE UNIQUE INDEX stand\_category\_uk ON standard\_category (stand\_id, cat\_id);

# Holds requirement information for each Category

CREATE TABLE IF NOT EXISTS requirement

(

req\_id INT NOT NULL AUTO\_INCREMENT,

req\_num VARCHAR(15) NOT NULL,

req\_desc TEXT NOT NULL, # Full description of requirement

req\_simple\_desc TEXT, # Shortened and easier to understand description for the requirement

req\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

req\_add\_uname VARCHAR(100),

req\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

req\_update\_uname VARCHAR(100),

standcat\_id INT NOT NULL,

PRIMARY KEY (req\_id),

FOREIGN KEY (standcat\_id) REFERENCES standard\_category (standcat\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to speed up retrievals of the requirement's number and description and make the pair unique

CREATE UNIQUE INDEX requirement\_uk ON requirement (req\_num, req\_desc(500));

# This table will hold newer verbiage for requirements of each standard

CREATE TABLE IF NOT EXISTS req\_amendment

(

amend\_id INT NOT NULL AUTO\_INCREMENT,

amend\_effective\_date DATE NOT NULL,

amend\_desc TEXT NOT NULL,

amend\_simple\_desc TEXT,

amend\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

amend\_add\_uname VARCHAR(100),

amend\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

amend\_update\_uname VARCHAR(100),

req\_id INT NOT NULL,

PRIMARY KEY (amend\_id),

FOREIGN KEY (req\_id) REFERENCES requirement (req\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to speed up retrieval of amendments to requirements

CREATE UNIQUE INDEX req\_amendment\_uk ON req\_amendment (req\_id, amend\_desc(500));

# Table will hold a limited number of possible ratings for results to each requirement a system is scanned against

CREATE TABLE IF NOT EXISTS rating

(

rate\_id INT NOT NULL AUTO\_INCREMENT,

rate\_name VARCHAR(25) NOT NULL,

rate\_abbv VARCHAR(5) NOT NULL,

rate\_desc VARCHAR(75) NOT NULL,

rate\_root\_stand INT NOT NULL,

rate\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

rate\_add\_uname VARCHAR(100),

rate\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

rate\_update\_uname VARCHAR(100),

PRIMARY KEY (rate\_id),

FOREIGN KEY (rate\_root\_stand) REFERENCES standard (stand\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE UNIQUE INDEX rating\_uk ON rating (rate\_abbv, rate\_root\_stand);

# This table will hold organization provided timelines for their estimated time to satisfy failed requirements

CREATE TABLE IF NOT EXISTS range\_time

(

range\_id INT NOT NULL AUTO\_INCREMENT,

range\_desc VARCHAR(50) NOT NULL,

range\_min INT NOT NULL,

range\_max INT NOT NULL,

range\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

range\_add\_uname VARCHAR(100),

range\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

range\_update\_uname VARCHAR(100),

PRIMARY KEY (range\_id)

) ENGINE=InnoDB;

# Neeed to speed up retrieval of data when requested by a query

CREATE UNIQUE INDEX range\_time\_uk ON range\_time (range\_min, range\_max);

# Table to hold all result information for each system that the requirement is tested against for each organization.

# Requirment/system pairs not in this table mean that that requirement has not been ran against that system yet.

CREATE TABLE IF NOT EXISTS system\_requirement

(

sysreq\_id INT NOT NULL AUTO\_INCREMENT,

sys\_id INT NOT NULL,

req\_id INT NOT NULL,

sysreq\_notes TEXT,

rate\_id INT NOT NULL,

range\_id INT,

art\_id INT NOT NULL,

sysreq\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

sysreq\_add\_uname VARCHAR(100),

sysreq\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

sysreq\_update\_uname VARCHAR(100),

PRIMARY KEY (sysreq\_id),

FOREIGN KEY (sys\_id) REFERENCES system (sys\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (req\_id) REFERENCES requirement (req\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (rate\_id) REFERENCES rating (rate\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (range\_id) REFERENCES range\_time (range\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (art\_id) REFERENCES artifact (art\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE UNIQUE INDEX system\_requirement\_uk ON system\_requirement (sys\_id, req\_id, sysreq\_add\_date);

# Table to store CSV loaded standard for processing into standard->category->standard\_category->requirement tables

CREATE TABLE standard\_import

(

standname varchar(100) NOT NULL,

standver varchar(10) NOT NULL,

standdate DATE NOT NULL,

catnum varchar(10) NOT NULL,

catname varchar(75) NOT NULL,

reqnum varchar(10) NOT NULL,

reqdesc text NOT NULL,

reqsimple text NOT NULL,

ratename varchar(25),

ratedesc varchar(75),

username varchar(100)

) ENGINE=InnoDB;

/\* Table to store CSV loaded results for processing into the SYSTEM\_REQUIREMENT

table \*/

CREATE TABLE result\_import

(

orgname VARCHAR(100),

sysname VARCHAR(100),

standname VARCHAR(100),

standver VARCHAR(10),

catnum VARCHAR(10),

reqnum INT,

artifact TEXT,

note TEXT,

ratename VARCHAR(25),

rangename VARCHAR(50),

username VARCHAR(100)

) ENGINE=InnoDB;

# Import standard results from csv

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `results\_import`()

BEGIN

# Remove trailing \r. It will get in the way......

SET SQL\_SAFE\_UPDATES = 0;

update result\_import SET rangename = TRIM(TRAILING '\r' FROM rangename);

SET SQL\_SAFE\_UPDATES=1;

INSERT INTO artifact (org\_id, art\_text, art\_add\_uname)

SELECT DISTINCT org\_id, artifact, username FROM result\_import ri JOIN organization ur ON ri.orgname = ur.org\_name;

INSERT INTO system\_requirement (sys\_id, req\_id, rate\_id, range\_id, art\_id, sysreq\_add\_uname)

SELECT DISTINCT sys\_id, req\_id, rate\_id, range\_id, art\_id, username from result\_import ri

JOIN system s ON ri.sysname=s.sys\_name

JOIN v\_standard vs ON ri.standname=vs.stand\_name AND ri.standver=vs.stand\_version\_rev\_num AND catnum=standcat\_num AND reqnum=req\_num

JOIN rating r ON ri.ratename=r.rate\_name AND (vs.stand\_root=r.rate\_root\_stand OR vs.stand\_id=r.rate\_root\_stand)

LEFT JOIN range\_time rt ON ri.rangename=rt.range\_desc

JOIN artifact a ON ri.artifact=a.art\_text;

TRUNCATE result\_import;

END$$

DELIMITER ;

/\* Table to store failed login attempts, when there are 5 within a two hour period the system will not allow the user to login \*/

CREATE TABLE login\_attempts

(

user\_id int(11) NOT NULL,

time varchar(30) NOT NULL

) ENGINE=InnoDB;

/\* Create Views \*/

CREATE OR REPLACE VIEW v\_user\_reference (

user\_email,

user\_id,

org\_id,

org\_name,

org\_mnged\_serv\_provider,

org\_parent\_branch,

sys\_id,

sys\_name

)

as

select

user.user\_email,

user.user\_id,

organization.org\_id,

organization.org\_name,

organization.org\_mnged\_serv\_provider,

organization.org\_parent\_branch,

system.sys\_id,

system.sys\_name

FROM organization LEFT JOIN system USING (org\_id)

JOIN user ON organization.org\_parent\_branch=user.org\_id OR organization.org\_id=user.org\_id

WHERE user\_id = getuser();

CREATE OR REPLACE VIEW v\_standard (

stand\_id,

stand\_name,

stand\_version\_rev\_num,

stand\_effective\_date,

stand\_root,

standcat\_id,

standcat\_num,

cat\_name,

req\_id,

req\_num,

req\_desc,

req\_simple\_desc

)

AS

SELECT

s.stand\_id,

s.stand\_name,

s.stand\_version\_rev\_num,

s.stand\_effective\_date,

s.stand\_root,

sc.standcat\_id,

sc.standcat\_num,

c.cat\_name,

r.req\_id,

r.req\_num,

r.req\_desc,

r.req\_simple\_desc

FROM standard s

JOIN standard\_category sc USING (stand\_id)

JOIN category c USING (cat\_id)

JOIN requirement r USING (standcat\_id)

ORDER BY stand\_name, stand\_version\_rev\_num, standcat\_id, req\_id;

CREATE OR REPLACE VIEW v\_system\_requirement (

sysreq\_id,

sys\_id,

req\_id,

sysreq\_add\_date,

sysreq\_notes,

rate\_id,

rate\_name,

range\_id,

range\_desc,

art\_id,

art\_text

)

AS

SELECT

sr.sysreq\_id,

sr.sys\_id,

sr.req\_id,

sr.sysreq\_add\_date,

sr.sysreq\_notes,

sr.rate\_id,

ra.rate\_name,

sr.range\_id,

rt.range\_desc,

sr.art\_id,

a.art\_text

FROM system\_requirement sr

JOIN rating ra USING (rate\_id)

LEFT JOIN range\_time rt USING (range\_id)

JOIN artifact a USING (art\_id)

WHERE sys\_id = getsys()

ORDER BY sys\_id, req\_id;

/\* Create Triggers \*/

DELIMITER $$

DROP TRIGGER IF EXISTS before\_standard\_insert//

CREATE TRIGGER before\_standard\_insert

BEFORE INSERT ON standard

FOR EACH ROW

BEGIN

DECLARE v\_stand INT;

SELECT MIN(stand\_id) FROM standard

WHERE stand\_name = NEW.stand\_name GROUP BY stand\_name

INTO v\_stand;

IF (v\_stand IS NOT NULL) THEN

SET NEW.stand\_root = v\_stand;

ELSE

SET NEW.stand\_root = NULL;

END IF;

END$$

/\* Create Functions \*/

CREATE FUNCTION getuser()

RETURNS INT(4)

RETURN @user$$

CREATE FUNCTION getsys()

RETURNS INT(4)

RETURN @sys$$

#Procedures

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_org\_sp`(IN oname VARCHAR(75), IN oaddr VARCHAR(100), IN ocity VARCHAR(60), IN ostate CHAR(2), IN otherstate CHAR(5), IN ozip VARCHAR(15), IN ocountry CHAR(5), IN parent INT(11), IN oprovider CHAR(1), IN uname VARCHAR(100))

BEGIN

INSERT INTO organization (org\_name, org\_address, org\_city, org\_state\_us, org\_state\_other, org\_zip, org\_country, org\_parent\_branch, org\_mnged\_serv\_provider, org\_add\_uname)

VALUES(oname, oaddr, ocity, ostate, otherstate, ozip, ocountry, parent, oprovider, uname);

END$$

# Insert system

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_system\_sp`(IN sysname VARCHAR(100), IN ipaddr VARCHAR(20), IN uname VARCHAR(100), IN systype INT, IN org INT)

BEGIN

INSERT INTO system (sys\_name, sys\_ip\_address, sys\_add\_uname, type\_id, org\_id)

VALUES(sysname, ipaddr, uname, systype, org);

END$$

# Insert system type

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_systype\_sp`(IN typename VARCHAR(75), IN uname VARCHAR(100))

BEGIN

INSERT INTO sys\_type (type\_name, type\_add\_uname)

VALUES(typename, uname);

END$$

# Insert new user

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_user\_sp`(IN fname VARCHAR(60), IN mname VARCHAR(60), IN lname VARCHAR(60), IN ccode CHAR(3), IN u\_phone VARCHAR(16), IN u\_email VARCHAR(100), IN u\_pass VARCHAR(255), IN uname VARCHAR(100), IN org INT, IN manager INT)

BEGIN

INSERT INTO user (user\_fname, user\_midname, user\_lname, user\_country\_code, user\_phone\_num, user\_email, user\_password, user\_add\_uname, org\_id, user\_manager)

VALUES(fname, mname, lname, ccode, u\_phone, u\_email, u\_pass, uname, org, manager);

END$$

# Import standard from csv

CREATE DEFINER=`root`@`localhost` PROCEDURE `standard\_import`()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE v\_reqnum INT;

DECLARE v\_standdate DATE;

DECLARE v\_reqdesc text;

DECLARE v\_reqsimple text;

DECLARE v\_username VARCHAR(100);

DECLARE cur1 CURSOR FOR SELECT standname, standver, catnum, catname, reqnum, standdate, reqdesc, reqsimple, username FROM standard\_import;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

# Start with standard (least granular); go to requirement (most granular)

INSERT IGNORE INTO standard (stand\_name, stand\_version\_rev\_num, stand\_effective\_date, stand\_add\_uname)

SELECT DISTINCT standname, standver, standdate, username FROM standard\_import;

INSERT IGNORE INTO category (cat\_name, cat\_add\_uname)

SELECT DISTINCT catname, username FROM standard\_import;

OPEN cur1;

read\_loop: LOOP

FETCH cur1 INTO v\_standname, v\_standver, v\_catnum, v\_catname, v\_reqnum, v\_standdate, v\_reqdesc, v\_reqsimple, v\_username;

IF done THEN

LEAVE read\_loop;

END IF;

# Will need standcat id for each requirement

INSERT IGNORE INTO standard\_category (stand\_id, cat\_id, standcat\_num, standcat\_add\_uname)

VALUES ((SELECT stand\_id FROM standard WHERE stand\_name = v\_standname AND stand\_version\_rev\_num = v\_standver),

(SELECT cat\_id FROM category WHERE cat\_name=v\_catname), v\_catnum, v\_username);

SELECT MAX(standcat\_id) INTO v\_standcat FROM standard\_category;

# Check if any standard, category, requirement combos in standard\_import exists already in requirement table

# Nested program to escape stupid bug for select...INTO tripping continue handler

BEGIN

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = FALSE;

SELECT req\_id INTO v\_oldreq FROM v\_standard

WHERE stand\_name = v\_standname

AND standcat\_num = v\_catnum

AND req\_num = v\_reqnum;

END;

IF v\_oldreq IS NOT NULL THEN

# Check if either description has been updated

SELECT req\_desc, req\_simple\_desc INTO v\_olddesc, v\_oldsimple FROM requirement

WHERE req\_id = v\_oldreq

AND req\_desc = v\_olddesc

AND req\_simple\_desc = v\_oldsimple;

# Include amendment if either has changed

IF v\_olddesc IS NOT NULL OR v\_oldsimple IS NOT NULL THEN

INSERT INTO req\_amendment (req\_id, amend\_effective\_date, amend\_desc, amend\_simple\_desc, amend\_add\_uname)

VALUES (v\_oldreq, v\_standdate, v\_reqdesc, v\_reqsimple, v\_username);

END IF;

# Either way, update stand\_cat id to most recent

UPDATE requirement SET standcat\_id = v\_standcat WHERE req\_id = v\_oldreq;

ELSE

INSERT INTO requirement (req\_num, req\_desc, req\_simple\_desc, standcat\_id, req\_add\_uname)

VALUES (v\_reqnum, v\_reqdesc, v\_reqsimple, v\_standcat, v\_username);

END IF;

END LOOP;

CLOSE cur1;

INSERT INTO rating (rate\_name, rate\_abbv, rate\_desc, rate\_root\_stand, rate\_add\_uname)

SELECT DISTINCT LEFT(ratename,LOCATE('-',ratename) - 1), SUBSTR(ratename,LOCATE('-',ratename) + 1),

ratedesc, IFNULL(stand\_root, stand\_id), v\_username

FROM standard\_import si JOIN standard s ON s.stand\_name=si.standname

WHERE ratename != '';

TRUNCATE standard\_import;

END$$

DELIMITER ;

/\* Insert test data \*/

INSERT INTO organization (org\_name, org\_address, org\_city, org\_state\_us, org\_state\_other, org\_zip, org\_country, org\_parent\_branch, org\_mnged\_serv\_provider)

VALUES

('TRG', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', NULL, 'Y'),

('Microsoft', '456 Dumb Road', 'Quebec City', NULL, 'QC', 98765,'CAN', NULL, 'Y'),

('Sony', '789 I Dont Know Where', 'Tokyo', NULL, 'TYO', 45612, 'JPN', NULL, 'Y'),

('G2 Opts', '205 Business Park Dr. #200', 'Virginia Beach', 'VA', NULL, 23462, 'USA', NULL, 'Y'),

('Nintendo', '111 Lena Arch', 'Norfolk', 'VA', NULL, 23518, 'USA', 1, 'N'),

('D Co.', '999 Fairy Lane', 'Detroit', 'MI', NULL, 78945, 'USA', 2, 'N'),

('Org 4', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 5', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 6', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N'),

('Org 7', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 8', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 9', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N'),

('Org 10', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 11', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 12', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N'),

('Org 13', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 14', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 15', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N'),

('Org 16', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 17', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 18', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N');

INSERT INTO user (user\_fname, user\_midname, user\_lname, user\_country\_code, user\_phone\_num, user\_email, user\_password, org\_id, user\_manager)

VALUES

('Jeff', 'Davis', 'Bauersfeld', 'USA', '757-374-7393', 'jeff@rebus.com', '$2y$10$65AhtlHnI7hSfa/3/L/UeeGVxujsXbUtTdbccpXd77RuElYPY/syS', 1, NULL),

('Christopher', 'James', 'Christopherson', 'BRZ', '1-555-213-4493', 'chris@rebus.com', '$2y$10$HUq/Lf1ALpMT6T5yOurF3OwW77fJ4x1Cs.2/O/rCwTsKnVzeixC4e', 2, NULL),

('Christine', 'Sam', 'Pants', 'USA', '999-654-1236', 'christine@rebus.com', '$2y$10$w3oKjM1GtuzqxWrkyh9rAOgbStA96n17xuZttJWUOCu7zeY5puq.K', 3, NULL),

('Zack', '', 'Cranston', 'USA', '123-456-7890', 'zack@rebus.com', '$2y$10$5GIQUSK5D7KAgsJqqJGgT.gELhAUroluNsBD/cgvwyk2Daq8XEH9e', 2, 2),

('Daniel', 'James', 'Bond', 'ENG', '1-555-555-5555', 'daniel@rebus.com', '$2y$10$Qgc//w.B63E0hH9WmzAoSuFJRCCVtFnUffyacc4cumr2MVlxNZy/6', 1, 1),

('Lemuel', 'A', 'Aaronson', 'ENG', '1-555-555-5555', 'lemuel@rebus.com', '$2y$10$68x.Czo10Qvqgp40ji6tMOP7BrBzM06T/tsYoMADBI3UeuGtdcXEK', 3, 3),

('Barry', 'B', 'Barrison', 'ENG', '1-555-555-5555', 'barry@gmail.com', 'james10hash', 1, 1),

('Curt', 'C', 'Curtson', 'ENG', '1-555-555-5555', 'curt@gmail.com', 'james10hash', 2, 2),

('Donald', 'D', 'Donaldson', 'ENG', '1-555-555-5555', 'donald@gmail.com', 'james10hash', 3, 3),

('Eric', 'E', 'Erickson', 'ENG', '1-555-555-5555', 'eric@gmail.com', 'james10hash', 1, 1),

('Frank', 'F', 'Frankison', 'USA', '757-374-7393', 'frank@gmail.com', 'jeff10hash', 1, 1),

('Greg', 'G', 'Gregson', 'BRZ', '1-555-213-4493', 'greg@gmail.com', 'chris10hash', 2, 2),

('Harold', 'H', 'Haroldson', 'USA', '999-654-1236', 'harold@gmail.com', 'miranda10hash', 3, 3),

('Ignis', 'I', 'Ignison', 'USA', '123-456-7890', 'ignis@gmail.com', 'brian10hash', 1, 1),

('James', 'J', 'Jamison', 'ENG', '1-555-555-5555', 'james@gmail.com', 'james10hash', 2, 2),

('Kelly', 'K', 'Kellison', 'ENG', '1-555-555-5555', 'kelly@gmail.com', 'james10hash', 3, 3),

('Larry', 'L', 'Larrison', 'ENG', '1-555-555-5555', 'larry@gmail.com', 'james10hash', 1, 1),

('Michael', 'M', 'Michaelson', 'ENG', '1-555-555-5555', 'michael@gmail.com', 'james10hash', 2, 2),

('Nicholas', 'N', 'Nicholson', 'ENG', '1-555-555-5555', 'nicholas@gmail.com', 'james10hash', 3, 3),

('Oscar', 'O', 'Oscarson', 'ENG', '1-555-555-5555', 'oscar@gmail.com', 'james10hash', 1, 1);

INSERT INTO sys\_type (type\_name)

VALUES

('server'),

('network appliance'),

('database'),

('workstation'),

('software');

INSERT INTO system (sys\_name, sys\_ip\_address, type\_id, org\_id)

VALUES

('server1', '12.13.14.15', 1, 1),

('switch1', '120.13.140.15', 2, 2),

('database1', '20.33.54.115', 3, 3),

('server2', '12.13.14.16', 1, 4),

('database2', '20.33.54.116', 3, 5),

('server3', '12.13.14.15', 1, 6),

('switch2', '120.13.140.15', 2, 2),

('database3', '20.33.54.115', 3, 3),

('server4', '12.13.14.16', 1, 1),

('database4', '20.33.54.116', 3, 4),

('server5', '12.13.14.15', 1, 5),

('switch3', '120.13.140.15', 2, 6),

('database5', '20.33.54.115', 3, 3),

('server6', '12.13.14.16', 1, 1),

('database6', '20.33.54.116', 3, 5),

('server7', '12.13.14.15', 1, 1),

('switch4', '120.13.140.15', 2, 4),

('database7', '20.33.54.115', 3, 3),

('server8', '12.13.14.16', 1, 1),

('database8', '20.33.54.116', 3, 2);

INSERT INTO artifact (art\_text, org\_id)

VALUES

('artifact1 text', 1),

('artifact2 text', 2),

('artifact3 text', 3),

('artifact4 text', 2),

('artifact5 text', 1),

('artifact6 text', 1),

('artifact7 text', 2),

('artifact8 text', 3),

('artifact9 text', 2),

('artifact10 text', 1),

('artifact11 text', 1),

('artifact12 text', 2),

('artifact13 text', 3),

('artifact14 text', 2),

('artifact15 text', 1),

('artifact16 text', 1),

('artifact17 text', 2),

('artifact18 text', 3),

('artifact19 text', 2),

('artifact20 text', 1);

INSERT INTO standard (stand\_name, stand\_version\_rev\_num, stand\_effective\_date)

VALUES

('stand1', '3.6', '2016-01-01'),

('stand2', '3.1.2', '2016-02-02'),

('stand3', '3.3', '2016-03-03'),

('stand1', '3.7', '2017-04-04'),

('stand2', '3.1.3', '2017-04-04');

INSERT INTO category (cat\_name)

VALUES

('category1'),

('category2'),

('category3'),

('category4'),

('category5'),

('category6'),

('category7'),

('category8'),

('category9'),

('category10'),

('category11'),

('category12');

INSERT INTO standard\_category (stand\_id, cat\_id, standcat\_num)

VALUES

(1, 1, 1.1),

(1, 2, 1.2),

(1, 3, 1.3),

(1, 4, 2.1),

(1, 5, 2.2),

(4, 1, 1.1),

(4, 2, 1.2),

(4, 3, 1.3),

(4, 4, 2.1),

(4, 5, 2.2),

(2, 6, 1.1),

(2, 7, 2.1),

(5, 6, 1.1),

(5, 7, 2.1),

(5, 8, 3.1),

(3, 9, 1.1),

(3, 10, 1.2);

INSERT INTO requirement (req\_num, req\_desc, req\_simple\_desc, standcat\_id)

VALUES

(1, 'Make stronger passwords', 'Password strength', 6),

(1, 'Install all the newest security patches', 'Install patches', 7),

(1, 'Make sure port 6000 is closed', 'Close ports', 8),

(2, 'Make sure all important data is backed up', 'Backup data', 8),

(1, 'Make sure folder permissions are set correctly', 'Fix permissions', 9),

(1, 'requirement 6', 'simple 6', 10),

(2, 'requirement 7', 'simple 7', 5),

(1, 'requirement 8', 'simple 8', 13),

(1, 'requirement 9', 'simple 9', 14),

(1, 'requirement 10', 'simple 10', 16),

(1, 'requirement 11', 'simple 11', 17),

(1, 'requirement 12', 'simple 12', 15);

INSERT INTO req\_amendment (amend\_effective\_date, amend\_desc, req\_id)

VALUES

(current\_date(), 'Make even stronger passwords', 1),

(current\_date(), 'Install even more patches', 2),

(current\_date(), 'Make sure port 6666 is closed now', 3),

(current\_date(), 'Make 3 backups (was 1)', 4),

(current\_date(), 'requirement 8 updated', 11),

(current\_date(), 'requirement 9 updated', 12);

INSERT INTO rating (rate\_name, rate\_abbv, rate\_desc, rate\_root\_stand)

VALUES

('Not Applicable', 'NA', 'Requirement doesn\'t apply to this system', 1),

('Not Compliant', 'NC', 'System fails requirement in all respects', 1),

('Partially Compliant', 'PC', 'System meets some parts of the requirement and fails other parts', 1),

('Fully Compliant', 'FC', 'System meets all parts of this requirement.', 1),

('Doesn\'t Apply', 'DA', 'Requirement doesn\'t apply to this system', 2),

('Open', 'O', 'System fails some or all of this requirement.', 2),

('Not Reviewed', 'NR', 'This requirements hasn\'t been reviewed yet for this system', 2),

('Not a Finding', 'NF', 'System meets all aspects of this requirement.', 2),

('Doesn\'t Apply', 'DA', 'Requirement doesn\'t apply to this system', 3),

('Open', 'O', 'System fails some or all of this requirement.', 3),

('Not Reviewed', 'NR', 'This requirements hasn\'t been reviewed yet for this system', 3),

('Not a Finding', 'NF', 'System meets all aspects of this requirement.', 3);

INSERT INTO range\_time (range\_desc, range\_min, range\_max)

VALUES

('Short Term', 0, 6),

('Medium Term', 7, 12),

('Long Term', 13, 24);

INSERT INTO system\_requirement (sys\_id, req\_id, sysreq\_notes, rate\_id, range\_id, art\_id)

VALUES

(1, 1, 'result1', 1, 1, 1),

(1, 2, 'result2', 2, 2, 2),

(1, 3, 'result3', 3, 3, 3),

(1, 4, 'result4', 4, 2, 4),

(1, 5, 'result5', 4, 2, 5),

(1, 6, 'result6', 6, NULL, 6),

(1, 7, 'result7', 1, 3, 7),

(2, 1, 'result8', 8, 3, 8),

(2, 4, 'result9', 9, 2, 9),

(2, 6, 'result10', 10, 2, 10),

(2, 8, 'result11', 10, 1, 11),

(2, 10, 'result12', 12, 3, 12),

(2, 11, 'result13', 1, 1, 13),

(2, 2, 'result14', 2, 1, 14),

(3, 3, 'result15', 3, 2, 15),

(3, 4, 'result16', 4, 1, 16),

(2, 9, 'result17', 10, 2, 11);

DROP DATABASE rebus;

CREATE DATABASE rebus;

USE rebus;

# Create Tables

# Table to hold all organizations

CREATE TABLE IF NOT EXISTS organization

(

org\_id INT NOT NULL AUTO\_INCREMENT,

org\_name VARCHAR(75) NOT NULL,

org\_address VARCHAR(100) NOT NULL,

org\_city VARCHAR(60) NOT NULL,

org\_state\_us CHAR(2),

org\_state\_other CHAR(5),

org\_zip VARCHAR(15) NOT NULL,

org\_country CHAR(5),

org\_parent\_branch INT,

org\_mnged\_serv\_provider ENUM('Y','N'),

org\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

org\_add\_uname VARCHAR(100),

org\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

org\_update\_uname VARCHAR(100),

PRIMARY KEY (org\_id),

FOREIGN KEY (org\_parent\_branch) REFERENCES organization (org\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

/\* Index to help speed up searches and make org\_name UNIQUE \*/

CREATE UNIQUE INDEX organization\_uk ON organization (org\_name);

CREATE TABLE IF NOT EXISTS user

(

user\_id INT NOT NULL AUTO\_INCREMENT,

user\_fname VARCHAR(60) NOT NULL,

user\_midname VARCHAR(60),

user\_lname VARCHAR(60) NOT NULL,

user\_country\_code CHAR(3),

user\_phone\_num VARCHAR(16) NOT NULL,

user\_email VARCHAR(100) NOT NULL, # this field servers as the person's username

/\* Passwords are hashed during user creation or password resets from website \*/

user\_password VARCHAR(255) NOT NULL,

user\_temp\_flag ENUM('Y', 'N') NOT NULL DEFAULT 'N',

user\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

user\_add\_uname VARCHAR(100),

user\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

user\_update\_uname VARCHAR(100),

org\_id INT,

user\_manager INT,

PRIMARY KEY (user\_id),

FOREIGN KEY (org\_id) REFERENCES organization (org\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (user\_manager) REFERENCES user (user\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Speeds to retrieval for logins and determining session information and makes user\_email UNIQUE

CREATE UNIQUE INDEX user\_uk ON user (user\_email);

# Table to hold all the different flavors of a system, should be a small table

CREATE TABLE IF NOT EXISTS sys\_type

(

type\_id INT NOT NULL AUTO\_INCREMENT,

type\_name VARCHAR(75) NOT NULL,

type\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

type\_add\_uname VARCHAR(100),

type\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

type\_update\_uname VARCHAR(100),

PRIMARY KEY (type\_id)

) ENGINE=InnoDB;

# Help speed retrieval as type\_name is used in certain queries and makes system type unique

CREATE UNIQUE INDEX sys\_type\_uk ON sys\_type (type\_name);

# Holds all information for each system an organization keeps scan results for

CREATE TABLE IF NOT EXISTS system

(

sys\_id INT NOT NULL AUTO\_INCREMENT,

sys\_name VARCHAR(100) NOT NULL,

sys\_ip\_address VARCHAR(20),

sys\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

sys\_add\_uname VARCHAR(100),

sys\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

sys\_update\_uname VARCHAR(100),

type\_id INT NOT NULL,

org\_id INT NOT NULL,

PRIMARY KEY (sys\_id),

FOREIGN KEY (type\_id) REFERENCES sys\_type (type\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (org\_id) REFERENCES organization (org\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Help speed up queries where a list of systems for each organization is needed and makes system names unique within a given organization

CREATE UNIQUE INDEX system\_uk ON system (org\_id, sys\_name);

# This table will be used to hold organization's reasonings for complying or not complying with a requirement

CREATE TABLE IF NOT EXISTS artifact

(

art\_id INT NOT NULL AUTO\_INCREMENT,

art\_text VARCHAR(255) NOT NULL,

art\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

art\_add\_uname VARCHAR(100),

art\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

art\_update\_uname VARCHAR(100),

org\_id INT NOT NULL,

PRIMARY KEY (art\_id),

FOREIGN KEY (org\_id) REFERENCES organization (org\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to speed up retrieval when entering results manually and set artifact text unique to the organization

CREATE UNIQUE INDEX artifact\_uk ON artifact (org\_id, art\_text);

# Table to hold all standards needed

CREATE TABLE IF NOT EXISTS standard

(

stand\_id INT NOT NULL AUTO\_INCREMENT,

stand\_name VARCHAR(100) NOT NULL,

stand\_version\_rev\_num VARCHAR(100) NOT NULL,

stand\_effective\_date DATE NOT NULL,

stand\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

stand\_add\_uname VARCHAR(100),

stand\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

stand\_update\_uname VARCHAR(100),

stand\_root INT,

PRIMARY KEY (stand\_id),

FOREIGN KEY (stand\_root) REFERENCES standard (stand\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to speed up queries where a standard name and revision number are needed and the pair is unique

CREATE UNIQUE INDEX standard\_uk ON standard (stand\_name, stand\_version\_rev\_num);

# Holds the categories for each standard, some categories can be used by more than one standard

CREATE TABLE IF NOT EXISTS category

(

cat\_id INT NOT NULL AUTO\_INCREMENT,

cat\_name VARCHAR(75) NOT NULL,

cat\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

cat\_add\_uname VARCHAR(100),

cat\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

cat\_update\_uname VARCHAR(100),

PRIMARY KEY (cat\_id)

) ENGINE=InnoDB;

# Needed to speed up retrieval of category names and numbers when needed and make the pair unique

CREATE UNIQUE INDEX category\_uk ON category (cat\_name);

# Bridge entity to account for the use of categories by multiple standards and their revisions

CREATE TABLE IF NOT EXISTS standard\_category

(

standcat\_id INT NOT NULL AUTO\_INCREMENT,

stand\_id INT NOT NULL,

cat\_id INT NOT NULL,

standcat\_num VARCHAR(10) NOT NULL,

standcat\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

standcat\_add\_uname VARCHAR(100),

standcat\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

standcat\_update\_uname VARCHAR(100),

PRIMARY KEY (standcat\_id),

FOREIGN KEY (stand\_id) REFERENCES standard (stand\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (cat\_id) REFERENCES category (cat\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to make sure each standard and category combination only occurs once

CREATE UNIQUE INDEX stand\_category\_uk ON standard\_category (stand\_id, cat\_id);

# Holds requirement information for each Category

CREATE TABLE IF NOT EXISTS requirement

(

req\_id INT NOT NULL AUTO\_INCREMENT,

req\_num VARCHAR(15) NOT NULL,

req\_desc TEXT NOT NULL, # Full description of requirement

req\_simple\_desc TEXT, # Shortened and easier to understand description for the requirement

req\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

req\_add\_uname VARCHAR(100),

req\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

req\_update\_uname VARCHAR(100),

standcat\_id INT NOT NULL,

PRIMARY KEY (req\_id),

FOREIGN KEY (standcat\_id) REFERENCES standard\_category (standcat\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to speed up retrievals of the requirement's number and description and make the pair unique

CREATE UNIQUE INDEX requirement\_uk ON requirement (req\_num, req\_desc(500));

# This table will hold newer verbiage for requirements of each standard

CREATE TABLE IF NOT EXISTS req\_amendment

(

amend\_id INT NOT NULL AUTO\_INCREMENT,

amend\_effective\_date DATE NOT NULL,

amend\_desc TEXT NOT NULL,

amend\_simple\_desc TEXT,

amend\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

amend\_add\_uname VARCHAR(100),

amend\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

amend\_update\_uname VARCHAR(100),

req\_id INT NOT NULL,

PRIMARY KEY (amend\_id),

FOREIGN KEY (req\_id) REFERENCES requirement (req\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

# Needed to speed up retrieval of amendments to requirements

CREATE UNIQUE INDEX req\_amendment\_uk ON req\_amendment (req\_id, amend\_desc(500));

# Table will hold a limited number of possible ratings for results to each requirement a system is scanned against

CREATE TABLE IF NOT EXISTS rating

(

rate\_id INT NOT NULL AUTO\_INCREMENT,

rate\_name VARCHAR(25) NOT NULL,

rate\_abbv VARCHAR(5) NOT NULL,

rate\_desc VARCHAR(75) NOT NULL,

rate\_root\_stand INT NOT NULL,

rate\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

rate\_add\_uname VARCHAR(100),

rate\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

rate\_update\_uname VARCHAR(100),

PRIMARY KEY (rate\_id),

FOREIGN KEY (rate\_root\_stand) REFERENCES standard (stand\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE UNIQUE INDEX rating\_uk ON rating (rate\_abbv, rate\_root\_stand);

# This table will hold organization provided timelines for their estimated time to satisfy failed requirements

CREATE TABLE IF NOT EXISTS range\_time

(

range\_id INT NOT NULL AUTO\_INCREMENT,

range\_desc VARCHAR(50) NOT NULL,

range\_min INT NOT NULL,

range\_max INT NOT NULL,

range\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

range\_add\_uname VARCHAR(100),

range\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

range\_update\_uname VARCHAR(100),

PRIMARY KEY (range\_id)

) ENGINE=InnoDB;

# Neeed to speed up retrieval of data when requested by a query

CREATE UNIQUE INDEX range\_time\_uk ON range\_time (range\_min, range\_max);

# Table to hold all result information for each system that the requirement is tested against for each organization.

# Requirment/system pairs not in this table mean that that requirement has not been ran against that system yet.

CREATE TABLE IF NOT EXISTS system\_requirement

(

sysreq\_id INT NOT NULL AUTO\_INCREMENT,

sys\_id INT NOT NULL,

req\_id INT NOT NULL,

sysreq\_notes TEXT,

rate\_id INT NOT NULL,

range\_id INT,

art\_id INT NOT NULL,

sysreq\_add\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

sysreq\_add\_uname VARCHAR(100),

sysreq\_update\_date DATETIME ON UPDATE CURRENT\_TIMESTAMP,

sysreq\_update\_uname VARCHAR(100),

PRIMARY KEY (sysreq\_id),

FOREIGN KEY (sys\_id) REFERENCES system (sys\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (req\_id) REFERENCES requirement (req\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (rate\_id) REFERENCES rating (rate\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (range\_id) REFERENCES range\_time (range\_id)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (art\_id) REFERENCES artifact (art\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE UNIQUE INDEX system\_requirement\_uk ON system\_requirement (sys\_id, req\_id, sysreq\_add\_date);

# Table to store CSV loaded standard for processing into standard->category->standard\_category->requirement tables

CREATE TABLE standard\_import

(

standname varchar(100) NOT NULL,

standver varchar(10) NOT NULL,

standdate DATE NOT NULL,

catnum varchar(10) NOT NULL,

catname varchar(75) NOT NULL,

reqnum varchar(10) NOT NULL,

reqdesc text NOT NULL,

reqsimple text NOT NULL,

ratename varchar(25),

ratedesc varchar(75),

username varchar(100)

) ENGINE=InnoDB;

/\* Table to store CSV loaded results for processing into the SYSTEM\_REQUIREMENT

table \*/

CREATE TABLE result\_import

(

orgname VARCHAR(100),

sysname VARCHAR(100),

standname VARCHAR(100),

standver VARCHAR(10),

catnum VARCHAR(10),

reqnum INT,

artifact TEXT,

note TEXT,

ratename VARCHAR(25),

rangename VARCHAR(50),

username VARCHAR(100)

) ENGINE=InnoDB;

# Import standard results from csv

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `results\_import`()

BEGIN

# Remove trailing \r. It will get in the way......

SET SQL\_SAFE\_UPDATES = 0;

update result\_import SET rangename = TRIM(TRAILING '\r' FROM rangename);

SET SQL\_SAFE\_UPDATES=1;

INSERT INTO artifact (org\_id, art\_text, art\_add\_uname)

SELECT DISTINCT org\_id, artifact, username FROM result\_import ri JOIN organization ur ON ri.orgname = ur.org\_name;

INSERT INTO system\_requirement (sys\_id, req\_id, rate\_id, range\_id, art\_id, sysreq\_add\_uname)

SELECT DISTINCT sys\_id, req\_id, rate\_id, range\_id, art\_id, username from result\_import ri

JOIN system s ON ri.sysname=s.sys\_name

JOIN v\_standard vs ON ri.standname=vs.stand\_name AND ri.standver=vs.stand\_version\_rev\_num AND catnum=standcat\_num AND reqnum=req\_num

JOIN rating r ON ri.ratename=r.rate\_name AND (vs.stand\_root=r.rate\_root\_stand OR vs.stand\_id=r.rate\_root\_stand)

LEFT JOIN range\_time rt ON ri.rangename=rt.range\_desc

JOIN artifact a ON ri.artifact=a.art\_text;

TRUNCATE result\_import;

END$$

DELIMITER ;

/\* Table to store failed login attempts, when there are 5 within a two hour period the system will not allow the user to login \*/

CREATE TABLE login\_attempts

(

user\_id int(11) NOT NULL,

time varchar(30) NOT NULL

) ENGINE=InnoDB;

/\* Create Views \*/

CREATE OR REPLACE VIEW v\_user\_reference (

user\_email,

user\_id,

org\_id,

org\_name,

org\_mnged\_serv\_provider,

org\_parent\_branch,

sys\_id,

sys\_name

)

as

select

user.user\_email,

user.user\_id,

organization.org\_id,

organization.org\_name,

organization.org\_mnged\_serv\_provider,

organization.org\_parent\_branch,

system.sys\_id,

system.sys\_name

FROM organization LEFT JOIN system USING (org\_id)

JOIN user ON organization.org\_parent\_branch=user.org\_id OR organization.org\_id=user.org\_id

WHERE user\_id = getuser();

CREATE OR REPLACE VIEW v\_standard (

stand\_id,

stand\_name,

stand\_version\_rev\_num,

stand\_effective\_date,

stand\_root,

standcat\_id,

standcat\_num,

cat\_name,

req\_id,

req\_num,

req\_desc,

req\_simple\_desc

)

AS

SELECT

s.stand\_id,

s.stand\_name,

s.stand\_version\_rev\_num,

s.stand\_effective\_date,

s.stand\_root,

sc.standcat\_id,

sc.standcat\_num,

c.cat\_name,

r.req\_id,

r.req\_num,

r.req\_desc,

r.req\_simple\_desc

FROM standard s

JOIN standard\_category sc USING (stand\_id)

JOIN category c USING (cat\_id)

JOIN requirement r USING (standcat\_id)

ORDER BY stand\_name, stand\_version\_rev\_num, standcat\_id, req\_id;

CREATE OR REPLACE VIEW v\_system\_requirement (

sysreq\_id,

sys\_id,

req\_id,

sysreq\_add\_date,

sysreq\_notes,

rate\_id,

rate\_name,

range\_id,

range\_desc,

art\_id,

art\_text

)

AS

SELECT

sr.sysreq\_id,

sr.sys\_id,

sr.req\_id,

sr.sysreq\_add\_date,

sr.sysreq\_notes,

sr.rate\_id,

ra.rate\_name,

sr.range\_id,

rt.range\_desc,

sr.art\_id,

a.art\_text

FROM system\_requirement sr

JOIN rating ra USING (rate\_id)

LEFT JOIN range\_time rt USING (range\_id)

JOIN artifact a USING (art\_id)

WHERE sys\_id = getsys()

ORDER BY sys\_id, req\_id;

/\* Create Triggers \*/

DELIMITER $$

DROP TRIGGER IF EXISTS before\_standard\_insert//

CREATE TRIGGER before\_standard\_insert

BEFORE INSERT ON standard

FOR EACH ROW

BEGIN

DECLARE v\_stand INT;

SELECT MIN(stand\_id) FROM standard

WHERE stand\_name = NEW.stand\_name GROUP BY stand\_name

INTO v\_stand;

IF (v\_stand IS NOT NULL) THEN

SET NEW.stand\_root = v\_stand;

ELSE

SET NEW.stand\_root = NULL;

END IF;

END$$

/\* Create Functions \*/

CREATE FUNCTION getuser()

RETURNS INT(4)

RETURN @user$$

CREATE FUNCTION getsys()

RETURNS INT(4)

RETURN @sys$$

#Procedures

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_org\_sp`(IN oname VARCHAR(75), IN oaddr VARCHAR(100), IN ocity VARCHAR(60), IN ostate CHAR(2), IN otherstate CHAR(5), IN ozip VARCHAR(15), IN ocountry CHAR(5), IN parent INT(11), IN oprovider CHAR(1), IN uname VARCHAR(100))

BEGIN

INSERT INTO organization (org\_name, org\_address, org\_city, org\_state\_us, org\_state\_other, org\_zip, org\_country, org\_parent\_branch, org\_mnged\_serv\_provider, org\_add\_uname)

VALUES(oname, oaddr, ocity, ostate, otherstate, ozip, ocountry, parent, oprovider, uname);

END$$

# Insert system

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_system\_sp`(IN sysname VARCHAR(100), IN ipaddr VARCHAR(20), IN uname VARCHAR(100), IN systype INT, IN org INT)

BEGIN

INSERT INTO system (sys\_name, sys\_ip\_address, sys\_add\_uname, type\_id, org\_id)

VALUES(sysname, ipaddr, uname, systype, org);

END$$

# Insert system type

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_systype\_sp`(IN typename VARCHAR(75), IN uname VARCHAR(100))

BEGIN

INSERT INTO sys\_type (type\_name, type\_add\_uname)

VALUES(typename, uname);

END$$

# Insert new user

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_user\_sp`(IN fname VARCHAR(60), IN mname VARCHAR(60), IN lname VARCHAR(60), IN ccode CHAR(3), IN u\_phone VARCHAR(16), IN u\_email VARCHAR(100), IN u\_pass VARCHAR(255), IN uname VARCHAR(100), IN org INT, IN manager INT)

BEGIN

INSERT INTO user (user\_fname, user\_midname, user\_lname, user\_country\_code, user\_phone\_num, user\_email, user\_password, user\_add\_uname, org\_id, user\_manager)

VALUES(fname, mname, lname, ccode, u\_phone, u\_email, u\_pass, uname, org, manager);

END$$

# Import standard from csv

CREATE DEFINER=`root`@`localhost` PROCEDURE `standard\_import`()

BEGIN

DECLARE done INT DEFAULT FALSE;

#DECLARE v\_oldstand INT;

DECLARE v\_oldreq INT;

DECLARE v\_olddesc text;

DECLARE v\_oldsimple text;

DECLARE v\_standcat INT;

DECLARE v\_standname VARCHAR(100);

DECLARE v\_standver VARCHAR(10);

DECLARE v\_catnum VARCHAR(10);

DECLARE v\_catname VARCHAR(75);

DECLARE v\_reqnum INT;

DECLARE v\_standdate DATE;

DECLARE v\_reqdesc text;

DECLARE v\_reqsimple text;

DECLARE v\_username VARCHAR(100);

DECLARE cur1 CURSOR FOR SELECT standname, standver, catnum, catname, reqnum, standdate, reqdesc, reqsimple, username FROM standard\_import;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

# Start with standard (least granular); go to requirement (most granular)

INSERT IGNORE INTO standard (stand\_name, stand\_version\_rev\_num, stand\_effective\_date, stand\_add\_uname)

SELECT DISTINCT standname, standver, standdate, username FROM standard\_import;

INSERT IGNORE INTO category (cat\_name, cat\_add\_uname)

SELECT DISTINCT catname, username FROM standard\_import;

OPEN cur1;

read\_loop: LOOP

FETCH cur1 INTO v\_standname, v\_standver, v\_catnum, v\_catname, v\_reqnum, v\_standdate, v\_reqdesc, v\_reqsimple, v\_username;

IF done THEN

LEAVE read\_loop;

END IF;

# Will need standcat id for each requirement

INSERT IGNORE INTO standard\_category (stand\_id, cat\_id, standcat\_num, standcat\_add\_uname)

VALUES ((SELECT stand\_id FROM standard WHERE stand\_name = v\_standname AND stand\_version\_rev\_num = v\_standver),

(SELECT cat\_id FROM category WHERE cat\_name=v\_catname), v\_catnum, v\_username);

SELECT MAX(standcat\_id) INTO v\_standcat FROM standard\_category;

# Check if any standard, category, requirement combos in standard\_import exists already in requirement table

# Nested program to escape stupid bug for select...INTO tripping continue handler

BEGIN

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = FALSE;

SELECT req\_id INTO v\_oldreq FROM v\_standard

WHERE stand\_name = v\_standname

AND standcat\_num = v\_catnum

AND req\_num = v\_reqnum;

END;

IF v\_oldreq IS NOT NULL THEN

# Check if either description has been updated

SELECT req\_desc, req\_simple\_desc INTO v\_olddesc, v\_oldsimple FROM requirement

WHERE req\_id = v\_oldreq

AND req\_desc = v\_olddesc

AND req\_simple\_desc = v\_oldsimple;

# Include amendment if either has changed

IF v\_olddesc IS NOT NULL OR v\_oldsimple IS NOT NULL THEN

INSERT INTO req\_amendment (req\_id, amend\_effective\_date, amend\_desc, amend\_simple\_desc, amend\_add\_uname)

VALUES (v\_oldreq, v\_standdate, v\_reqdesc, v\_reqsimple, v\_username);

END IF;

# Either way, update stand\_cat id to most recent

UPDATE requirement SET standcat\_id = v\_standcat WHERE req\_id = v\_oldreq;

ELSE

INSERT INTO requirement (req\_num, req\_desc, req\_simple\_desc, standcat\_id, req\_add\_uname)

VALUES (v\_reqnum, v\_reqdesc, v\_reqsimple, v\_standcat, v\_username);

END IF;

END LOOP;

CLOSE cur1;

INSERT INTO rating (rate\_name, rate\_abbv, rate\_desc, rate\_root\_stand, rate\_add\_uname)

SELECT DISTINCT LEFT(ratename,LOCATE('-',ratename) - 1), SUBSTR(ratename,LOCATE('-',ratename) + 1),

ratedesc, IFNULL(stand\_root, stand\_id), v\_username

FROM standard\_import si JOIN standard s ON s.stand\_name=si.standname

WHERE ratename != '';

TRUNCATE standard\_import;

END$$

DELIMITER ;

/\* Insert test data \*/

INSERT INTO organization (org\_name, org\_address, org\_city, org\_state\_us, org\_state\_other, org\_zip, org\_country, org\_parent\_branch, org\_mnged\_serv\_provider)

VALUES

('TRG', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', NULL, 'Y'),

('Microsoft', '456 Dumb Road', 'Quebec City', NULL, 'QC', 98765,'CAN', NULL, 'Y'),

('Sony', '789 I Dont Know Where', 'Tokyo', NULL, 'TYO', 45612, 'JPN', NULL, 'Y'),

('G2 Opts', '205 Business Park Dr. #200', 'Virginia Beach', 'VA', NULL, 23462, 'USA', NULL, 'Y'),

('Nintendo', '111 Lena Arch', 'Norfolk', 'VA', NULL, 23518, 'USA', 1, 'N'),

('D Co.', '999 Fairy Lane', 'Detroit', 'MI', NULL, 78945, 'USA', 2, 'N'),

('Org 4', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 5', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 6', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N'),

('Org 7', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 8', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 9', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N'),

('Org 10', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 11', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 12', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N'),

('Org 13', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 14', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 15', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N'),

('Org 16', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 3, 'N'),

('Org 17', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 1, 'N'),

('Org 18', '123 Fake Street', 'San Francisco', 'CA', NULL, 12345, 'USA', 2, 'N');

INSERT INTO user (user\_fname, user\_midname, user\_lname, user\_country\_code, user\_phone\_num, user\_email, user\_password, org\_id, user\_manager)

VALUES

('Jeff', 'Davis', 'Bauersfeld', 'USA', '757-374-7393', 'jeff@rebus.com', '$2y$10$65AhtlHnI7hSfa/3/L/UeeGVxujsXbUtTdbccpXd77RuElYPY/syS', 1, NULL),

('Christopher', 'James', 'Christopherson', 'BRZ', '1-555-213-4493', 'chris@rebus.com', '$2y$10$HUq/Lf1ALpMT6T5yOurF3OwW77fJ4x1Cs.2/O/rCwTsKnVzeixC4e', 2, NULL),

('Christine', 'Sam', 'Pants', 'USA', '999-654-1236', 'christine@rebus.com', '$2y$10$w3oKjM1GtuzqxWrkyh9rAOgbStA96n17xuZttJWUOCu7zeY5puq.K', 3, NULL),

('Zack', '', 'Cranston', 'USA', '123-456-7890', 'zack@rebus.com', '$2y$10$5GIQUSK5D7KAgsJqqJGgT.gELhAUroluNsBD/cgvwyk2Daq8XEH9e', 2, 2),

('Daniel', 'James', 'Bond', 'ENG', '1-555-555-5555', 'daniel@rebus.com', '$2y$10$Qgc//w.B63E0hH9WmzAoSuFJRCCVtFnUffyacc4cumr2MVlxNZy/6', 1, 1),

('Lemuel', 'A', 'Aaronson', 'ENG', '1-555-555-5555', 'lemuel@rebus.com', '$2y$10$68x.Czo10Qvqgp40ji6tMOP7BrBzM06T/tsYoMADBI3UeuGtdcXEK', 3, 3),

('Barry', 'B', 'Barrison', 'ENG', '1-555-555-5555', 'barry@gmail.com', 'james10hash', 1, 1),

('Curt', 'C', 'Curtson', 'ENG', '1-555-555-5555', 'curt@gmail.com', 'james10hash', 2, 2),

('Donald', 'D', 'Donaldson', 'ENG', '1-555-555-5555', 'donald@gmail.com', 'james10hash', 3, 3),

('Eric', 'E', 'Erickson', 'ENG', '1-555-555-5555', 'eric@gmail.com', 'james10hash', 1, 1),

('Frank', 'F', 'Frankison', 'USA', '757-374-7393', 'frank@gmail.com', 'jeff10hash', 1, 1),

('Greg', 'G', 'Gregson', 'BRZ', '1-555-213-4493', 'greg@gmail.com', 'chris10hash', 2, 2),

('Harold', 'H', 'Haroldson', 'USA', '999-654-1236', 'harold@gmail.com', 'miranda10hash', 3, 3),

('Ignis', 'I', 'Ignison', 'USA', '123-456-7890', 'ignis@gmail.com', 'brian10hash', 1, 1),

('James', 'J', 'Jamison', 'ENG', '1-555-555-5555', 'james@gmail.com', 'james10hash', 2, 2),

('Kelly', 'K', 'Kellison', 'ENG', '1-555-555-5555', 'kelly@gmail.com', 'james10hash', 3, 3),

('Larry', 'L', 'Larrison', 'ENG', '1-555-555-5555', 'larry@gmail.com', 'james10hash', 1, 1),

('Michael', 'M', 'Michaelson', 'ENG', '1-555-555-5555', 'michael@gmail.com', 'james10hash', 2, 2),

('Nicholas', 'N', 'Nicholson', 'ENG', '1-555-555-5555', 'nicholas@gmail.com', 'james10hash', 3, 3),

('Oscar', 'O', 'Oscarson', 'ENG', '1-555-555-5555', 'oscar@gmail.com', 'james10hash', 1, 1);

INSERT INTO sys\_type (type\_name)

VALUES

('server'),

('network appliance'),

('database'),

('workstation'),

('software');

INSERT INTO system (sys\_name, sys\_ip\_address, type\_id, org\_id)

VALUES

('server1', '12.13.14.15', 1, 1),

('switch1', '120.13.140.15', 2, 2),

('database1', '20.33.54.115', 3, 3),

('server2', '12.13.14.16', 1, 4),

('database2', '20.33.54.116', 3, 5),

('server3', '12.13.14.15', 1, 6),

('switch2', '120.13.140.15', 2, 2),

('database3', '20.33.54.115', 3, 3),

('server4', '12.13.14.16', 1, 1),

('database4', '20.33.54.116', 3, 4),

('server5', '12.13.14.15', 1, 5),

('switch3', '120.13.140.15', 2, 6),

('database5', '20.33.54.115', 3, 3),

('server6', '12.13.14.16', 1, 1),

('database6', '20.33.54.116', 3, 5),

('server7', '12.13.14.15', 1, 1),

('switch4', '120.13.140.15', 2, 4),

('database7', '20.33.54.115', 3, 3),

('server8', '12.13.14.16', 1, 1),

('database8', '20.33.54.116', 3, 2);

INSERT INTO artifact (art\_text, org\_id)

VALUES

('artifact1 text', 1),

('artifact2 text', 2),

('artifact3 text', 3),

('artifact4 text', 2),

('artifact5 text', 1),

('artifact6 text', 1),

('artifact7 text', 2),

('artifact8 text', 3),

('artifact9 text', 2),

('artifact10 text', 1),

('artifact11 text', 1),

('artifact12 text', 2),

('artifact13 text', 3),

('artifact14 text', 2),

('artifact15 text', 1),

('artifact16 text', 1),

('artifact17 text', 2),

('artifact18 text', 3),

('artifact19 text', 2),

('artifact20 text', 1);

INSERT INTO standard (stand\_name, stand\_version\_rev\_num, stand\_effective\_date)

VALUES

('stand1', '3.6', '2016-01-01'),

('stand2', '3.1.2', '2016-02-02'),

('stand3', '3.3', '2016-03-03'),

('stand1', '3.7', '2017-04-04'),

('stand2', '3.1.3', '2017-04-04');

INSERT INTO category (cat\_name)

VALUES

('category1'),

('category2'),

('category3'),

('category4'),

('category5'),

('category6'),

('category7'),

('category8'),

('category9'),

('category10'),

('category11'),

('category12');

INSERT INTO standard\_category (stand\_id, cat\_id, standcat\_num)

VALUES

(1, 1, 1.1),

(1, 2, 1.2),

(1, 3, 1.3),

(1, 4, 2.1),

(1, 5, 2.2),

(4, 1, 1.1),

(4, 2, 1.2),

(4, 3, 1.3),

(4, 4, 2.1),

(4, 5, 2.2),

(2, 6, 1.1),

(2, 7, 2.1),

(5, 6, 1.1),

(5, 7, 2.1),

(5, 8, 3.1),

(3, 9, 1.1),

(3, 10, 1.2);

INSERT INTO requirement (req\_num, req\_desc, req\_simple\_desc, standcat\_id)

VALUES

(1, 'Make stronger passwords', 'Password strength', 6),

(1, 'Install all the newest security patches', 'Install patches', 7),

(1, 'Make sure port 6000 is closed', 'Close ports', 8),

(2, 'Make sure all important data is backed up', 'Backup data', 8),

(1, 'Make sure folder permissions are set correctly', 'Fix permissions', 9),

(1, 'requirement 6', 'simple 6', 10),

(2, 'requirement 7', 'simple 7', 5),

(1, 'requirement 8', 'simple 8', 13),

(1, 'requirement 9', 'simple 9', 14),

(1, 'requirement 10', 'simple 10', 16),

(1, 'requirement 11', 'simple 11', 17),

(1, 'requirement 12', 'simple 12', 15);

INSERT INTO req\_amendment (amend\_effective\_date, amend\_desc, req\_id)

VALUES

(current\_date(), 'Make even stronger passwords', 1),

(current\_date(), 'Install even more patches', 2),

(current\_date(), 'Make sure port 6666 is closed now', 3),

(current\_date(), 'Make 3 backups (was 1)', 4),

(current\_date(), 'requirement 8 updated', 11),

(current\_date(), 'requirement 9 updated', 12);

INSERT INTO rating (rate\_name, rate\_abbv, rate\_desc, rate\_root\_stand)

VALUES

('Not Applicable', 'NA', 'Requirement doesn\'t apply to this system', 1),

('Not Compliant', 'NC', 'System fails requirement in all respects', 1),

('Partially Compliant', 'PC', 'System meets some parts of the requirement and fails other parts', 1),

('Fully Compliant', 'FC', 'System meets all parts of this requirement.', 1),

('Doesn\'t Apply', 'DA', 'Requirement doesn\'t apply to this system', 2),

('Open', 'O', 'System fails some or all of this requirement.', 2),

('Not Reviewed', 'NR', 'This requirements hasn\'t been reviewed yet for this system', 2),

('Not a Finding', 'NF', 'System meets all aspects of this requirement.', 2),

('Doesn\'t Apply', 'DA', 'Requirement doesn\'t apply to this system', 3),

('Open', 'O', 'System fails some or all of this requirement.', 3),

('Not Reviewed', 'NR', 'This requirements hasn\'t been reviewed yet for this system', 3),

('Not a Finding', 'NF', 'System meets all aspects of this requirement.', 3);

INSERT INTO range\_time (range\_desc, range\_min, range\_max)

VALUES

('Short Term', 0, 6),

('Medium Term', 7, 12),

('Long Term', 13, 24);

INSERT INTO system\_requirement (sys\_id, req\_id, sysreq\_notes, rate\_id, range\_id, art\_id)

VALUES

(1, 1, 'result1', 1, 1, 1),

(1, 2, 'result2', 2, 2, 2),

(1, 3, 'result3', 3, 3, 3),

(1, 4, 'result4', 4, 2, 4),

(1, 5, 'result5', 4, 2, 5),

(1, 6, 'result6', 6, NULL, 6),

(1, 7, 'result7', 1, 3, 7),

(2, 1, 'result8', 8, 3, 8),

(2, 4, 'result9', 9, 2, 9),

(2, 6, 'result10', 10, 2, 10),

(2, 8, 'result11', 10, 1, 11),

(2, 10, 'result12', 12, 3, 12),

(2, 11, 'result13', 1, 1, 13),

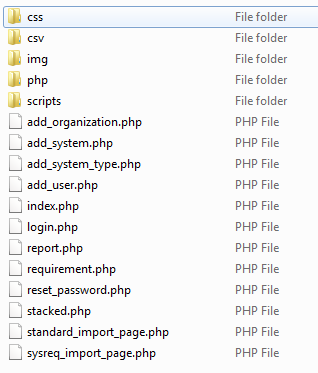
(2, 2, 'result14', 2, 1, 14),

(3, 3, 'result15', 3, 2, 15),

(3, 4, 'result16', 4, 1, 16),

(2, 9, 'result17', 10, 2, 11);

**Description of the files**



**Folders:**

* CSS – holds all of the style sheets for the website
* CSV – holds the CSV’s used during testing
* Img – holds all of the images used on the website
* PHP – holds all of the action files that the web files in the above screenshot call when a submit button or change on the webpage is made
* Scripts – holds all of the JQuery and Javascript files used by the website.

**Web Files:**

* Add\_organization.php – holds the code for the page you see when you click the Add Organization link on the website. When the submit button is clicked, the php/insert\_organization.php file is called.
* Add\_system.php – holds the code for the page when clicking the Add System link on the website. When the submit button is clicked, the php/insert\_system.php file is called.
* Add\_system\_type.php – holds the code for the page when clicking the Add System Type link on the website. When the submit button is clicked, the php/insert\_system\_type.php file is called.
* Add\_user.php - holds the code for the page when clicking the Add User link on the website. When the submit button is clicked, the php/insert\_user.php file is called.
* Index.php – This file holds the code for the landing page after a user successfully logs in. It calls the scripts/functions that display the charts and the ribbon bar with links to all the other pages.
* Login.php – This holds the code for the initial log on page for the website and calls the php/login.php file for processing the user’s input.
* Report.php – This holds the code to display the reporting page
* Requirement.php – Code for user inserted requirements
* Reset\_password.php – This holds the code to display the reset password page and calls the php/reset.php file for processing the user’s input.
* Stacked.php – This also holds code for the charts on the main splash page
* Standard\_import\_page.php - holds the code for the page you see when you click the Import Standard link on the website. When the submit button is clicked, the php/standard\_import.php file is called.
* Sysreq\_import\_page.php - holds the code for the page you see when you click the Import Results link on the website. When the submit button is clicked, the php/sysreq\_import.php file is called.

**Log-In Authentication System:**

* User navigates to log-in page (trg\_login.html)
* User enters ID (e-mail address) and password
* Information is sent to log\_in.php which accepts t and sends it to the functions.php
* The functions.php compares the userID to the authorized user information in the USER table
* The password is hashed and compared to the password in the USER table
* If userID and password match, the user is able to proceed with the log-in to the dashboard
* If the log-in information does not match, the user will see a message that the credentials are not valid
* The authentication system will log failed attempts by inserting the userID from the USER table and time into a LOGIN\_ATTEMPTS table.
* After 5 failed log-in attempts within a two-hour period, the user access is denied and will be locked out
* To recover a forgotten password, the user can click the Forgot Password link and type in the email address to have temporary password emailed.

<?php

include\_once('php/functions.php');

sec\_session\_start();

?>

<html>

<head>

<title>Reset Password</title>

<meta charset="UTF-8" />

<meta name"author" content="Rebus Group" />

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<link href="css/styles.css" rel="stylesheet" type="text/css" />

</head>

<body>

<div class="container" id="prefooter">

<!--Menu bar -->

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<div class="navbar-header">

<img id="title" src="img/G2globe-gray.png" style="height:50px;"/>

</div>

<ul class="nav navbar-nav">

<li><a href="index.php" target="\_self">Home</a></li>

<li><a href="add\_organization.php" target="\_self">Add Organization</a></li>

<li><a href="add\_user.php" target="\_self">Add User</a></li>

<li><a href="add\_system.php" target="\_self">Add System</a></li>

<li><a href="add\_system\_type.php" target="\_self">Add System Type</a></li>

<li><a href="report.php" target="\_self">Report</a></li>

<li><a href="sysreq\_import\_page.php" target="\_self">Import</a></li>

<li><a href="php/logout.php" target="\_self">Logout</a></li>

</ul>

</div>

</nav>

<h3><?php if (isset($\_SESSION["message"])) { echo $\_SESSION["message"]; unset($\_SESSION["message"]); } ?></h3>

<h2>Reset Password</h2>

<div class="text-center">

<p>You will receive an email with your reset password. After logging in with that passoword, you will be prompted to put in a new, permanent password.</p>

<form method="post" action="php/reset.php">

<div class="form-group">

<input type="email" name="useremail" placeholder="email">

</div>

<button type="submit" class="btn btn-primary">Reset</button>

</form>

* Upon logging-in with temporary password, the user will be prompted to choose a permanent password

<?php

session\_name("sec\_session\_id");

session\_start();

if (isset($\_SESSION["user\_id"]) && $\_SESSION["expire"] < time()) {

header("Location: index.php");

}

?>

<!--Login Page of the Tidewater Rebus Group-->

<html>

<head>

<title>Login</title>

<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css"/>

<link href="css/styles.css" rel="stylesheet" type="text/css" />

</head>

<body>

<div class="container text-center align-middle" id="prefooter">

<h3 id="error">

<?php

// Login Error

if (isset($\_GET["error"])) {

// Invalid Credentials

if ($\_GET["error"] == 1) {

echo 'You have provided invalid credentials.';

}

// The correct POST variables were not sent to this page.

else if ($\_GET["error"] == 2) {

echo 'Invalid Request. Contact administrator for assistance.';

}

else if ($\_GET["error"] == 3) {

echo 'You are not authorized to visit this page. You have been logged out.';

}

}

?>

</h3>

<div id="login" style="padding-top: 100px">

<h2>Login</h2>

<form action="php/login.php" method="POST">

<div class="form-group">

<input type="email" name="username" placeholder="Email" autofocus required></br>

</div>

<div class="form-group">

<input type="password" name="password" placeholder="Password" required><br/>

</div>

<button type="submit" class="btn btn-primary">Log In</button>

</form>

</div>

<div id="forgot" style="padding-top: 20px">

<a href="reset\_page.php">Forgot Password?</a>

</div>

</div>

<div class="container" id="footer">

<footer>

205 Business Park Dr #200 &diams;

Virginia Beach, VA 23462 &diams;

(757) 965-8330 &diams;

</footer>

</div>

</body>

</html>

* To navigate to other pages without valid log-in is prohibited and the user will be redirected to the log-in page
* User can disconnect by using a logout.php file

**Security:**

If a user attempts to arrive at any page other than the login page without having logged in ($\_SESSION[“user\_id”] will not be set), he will be redirected to the login page. All filters are dropdown menus to minimize security risk and user error.

**Functions:**

All report.php functions utilize jQuery and AJAX calls. They are linked to report.php from scripts/rep\_scripts.js. Functions are as follows:

<?php

include ("connect/connect.php");

// Authentication

function sec\_session\_start() {

$session\_name = 'sec\_session\_id'; // Set a custom session name

$secure = false; // Set to true is using https

// This stops JavaScript being able to access the session id.

$httponly = true;

// Forces sessions to only use cookies.

if (ini\_set('session.use\_only\_cookies', 1) === FALSE) {

header("Location: ../error.php?err=Could not initiate a safe session (ini\_set)");

exit();

}

// Gets current cookies params.

$cookieParams = session\_get\_cookie\_params();

session\_set\_cookie\_params($cookieParams["lifetime"],

$cookieParams["path"],

$cookieParams["domain"],

$secure,

$httponly);

// Sets the session name to the one set above.

session\_name($session\_name);

session\_start(); // Start the PHP session

session\_regenerate\_id(true); // regenerated the session, delete the old one.

}

// Determines if login attempt is successful or not

function login($email, $password, $conn) {

// Using prepared statements means that SQL injection is not possible.

if ($stmt = $conn->prepare("SELECT user\_id, user\_email, user\_password

FROM user

WHERE user\_email = ?

LIMIT 1")) {

$stmt->bind\_param('s', $email); // Bind "$email" to parameter.

$stmt->execute(); // Execute the prepared query.

$stmt->store\_result();

// get variables from result.

$stmt->bind\_result($user\_id, $username, $db\_password);

$stmt->fetch();

if ($stmt->num\_rows == 1) {

// If the user exists we check if the account is locked

// from too many login attempts

if (checkbrute($user\_id, $conn) == true) {

// Account is locked

// Send an email to user saying their account is locked

return false;

} else {

// Check if the password in the database matches

// the password the user submitted. We are using

// the password\_verify function to avoid timing attacks.

if (password\_verify($password, $db\_password)) {

// Password is correct!

// Get the user-agent string of the user.

$user\_browser = $\_SERVER['HTTP\_USER\_AGENT'];

// XSS protection as we might print this value

$user\_id = preg\_replace("/[^0-9]+/", "", $user\_id);

$\_SESSION['user\_id'] = $user\_id;

// XSS protection as we might print this value

$username = preg\_replace("/[^a-zA-Z0-9\_\-@.]+/",

"",

$username);

$\_SESSION['username'] = $username;

$\_SESSION['login\_string'] = hash('sha512',

$db\_password . $user\_browser);

$\_SESSION['expire'] = time() + 900;

// Login successful.

return true;

} else {

// Password is not correct

// We record this attempt in the database

$now = time();

$conn->query("INSERT INTO login\_attempts(user\_id, time)

VALUES ('$user\_id', '$now')");

return false;

}

}

} else {

// No user exists.

return false;

}

}

}

// For logging failed login attempts

function checkbrute($user\_id, $conn) {

// Get timestamp of current time

$now = time();

// All login attempts are counted from the past 2 hours.

$valid\_attempts = $now - (2 \* 60 \* 60);

if ($stmt = $conn->prepare("SELECT time

FROM login\_attempts

WHERE user\_id = ?

AND time > '$valid\_attempts'")) {

$stmt->bind\_param('i', $user\_id);

// Execute the prepared query.

$stmt->execute();

$stmt->store\_result();

// If there have been more than 5 failed logins

if ($stmt->num\_rows > 5) {

return true;

} else {

return false;

}

}

}

function login\_check($conn) {

// Check if all session variables are set and last activity was < 15 minutes ago

if (isset($\_SESSION['user\_id'],

$\_SESSION['username'],

$\_SESSION['login\_string'])

&& time() < $\_SESSION['expire']) {

$user\_id = $\_SESSION['user\_id'];

$login\_string = $\_SESSION['login\_string'];

$username = $\_SESSION['username'];

// Get the user-agent string of the user.

$user\_browser = $\_SERVER['HTTP\_USER\_AGENT'];

if ($stmt = $conn->prepare("SELECT user\_password

FROM user

WHERE user\_id = ? LIMIT 1")) {

// Bind "$user\_id" to parameter.

$stmt->bind\_param('i', $user\_id);

$stmt->execute(); // Execute the prepared query.

$stmt->store\_result();

if ($stmt->num\_rows == 1) {

// If the user exists get variables from result.

$stmt->bind\_result($password);

$stmt->fetch();

$login\_check = hash('sha512', $password . $user\_browser);

if (hash\_equals($login\_check, $login\_string) ){

// Logged In!!!!

return true;

} else {

// Not logged in

return false;

}

} else {

// Not logged in

return false;

}

} else {

// Not logged in

return false;

}

} else {

// Not logged in

return false;

}

}

// Processes a user logout and kills all sessions

function logout($loc) {

// Unset all session values

$\_SESSION = array();

// get session parameters

$params = session\_get\_cookie\_params();

// Delete the actual cookie.

setcookie(session\_name(),

'', time() - 900,

$params["path"],

$params["domain"],

$params["secure"],

$params["httponly"]);

// Destroy session

session\_unset();

session\_destroy();

if ($loc == "web") {

header('Location: login.php?error=0');

}

else if ($loc == "php") {

header('Location: ../login.php?error=0');

}

else if ($loc == "unauthweb") {

header('Location: login.php?error=3');

}

else if ($loc == "unauthphp") {

header('Location: ../login.php?error=3');

}

}

// Data Manipulation/Expression

// Return organizations

function get\_orgs($userid, $conn) {

include ("connect/connect.php");

// Set user session variable

$sql = 'SET @user = "'.$userid.'"';

mysqli\_query($conn, $sql) or die(mysqli\_error($conn));

// Gather user's organizations

$sql = 'SELECT DISTINCT org\_name, org\_id FROM v\_user\_reference';

$result = mysqli\_query($conn, $sql) or die(mysqli\_error($conn));

mysqli\_close($conn);

return $result;

}

// Returns system types

function get\_systype($conn) {

$sql = 'SELECT type\_id, type\_name FROM sys\_type';

$result = mysqli\_query($conn, $sql) or die(mysqli\_error($conn));

mysqli\_close($conn);

return $result;

}

// Creates Requirement Detail page for selected requirement from Report page

function load\_requirement($req, $sys, $conn) {

// Set system session variable

$stmt = $conn->prepare('SET @sys = ?');

$stmt->bind\_param("i", $sys);

$sys = $\_GET["sys"];

$stmt->execute();

// Regather basic identifying information of the requirement

$stmt = $conn->prepare('SELECT stand\_name, stand\_version\_rev\_num, standcat\_num, cat\_name, req\_num, req\_desc, req\_simple\_desc FROM v\_standard WHERE req\_id = ?');

$stmt->bind\_param("i", $req);

$req = $\_GET["req"];

$stmt->execute();

$result = $stmt->get\_result();

$reqrow = $result->fetch\_assoc();

// Gather each time this requirement was run

$stmt = $conn->prepare('SELECT sysreq\_id, sysreq\_add\_date FROM v\_system\_requirement WHERE req\_id = ? ORDER BY sysreq\_id DESC');

$stmt->bind\_param("i", $req);

$req = $\_GET["req"];

$stmt->execute();

$sysreqresult = $stmt->get\_result();

// Gather results; potentially multiple entries for system-requirement pair

while ($row = $sysreqresult->fetch\_assoc()) {

$sysreqs[] = array("id"=>$row["sysreq\_id"], "date"=>$row["sysreq\_add\_date"]);

}

if ($sysreqresult->num\_rows > 0) {

// Gather result details (default to most recent)

$stmt = $conn->prepare('SELECT sysreq\_notes, rate\_id, range\_id, range\_desc, art\_id, art\_text FROM v\_system\_requirement WHERE sysreq\_id = ?');

$stmt->bind\_param("i", $sysreq);

$sysreq = $sysreqs[0]["id"];

$stmt->execute();

$result = $stmt->get\_result();

$selectedrow = $result->fetch\_assoc();

}

// Gather ratings for standard

$stmt = $conn->prepare('SELECT rate\_id, rate\_abbv, rate\_name FROM rating WHERE rate\_root\_stand = ? OR rate\_root\_stand = (SELECT stand\_root FROM standard WHERE stand\_id = ?)');

$stmt->bind\_param("ii", $stand, $stand);

$stand = $\_GET["stand"];

$stmt->execute();

$rateresult = $stmt->get\_result();

mysqli\_close($conn);

// Begin Details row and "changeable details" div

echo '

<h2>Requirement Details: '.$reqrow["standcat\_num"].'.'.$reqrow["req\_num"].'</h2>

<div id="details\_row" class="row">

<div id="changeable\_details" class="col-md-6">';

if (isset($selectedrow)) {

// Dropdown of results that will populate with corresponding date's result information when clicked

echo '

<label>Result Date</label>

<select id="sysreq\_dropdown">';

foreach ($sysreqs as $sysreq=>$array) {

echo '

<option value = "'.$array["id"].'">'.$array["date"].'</option>';

}

echo '

</select>

<form action = "" method ="post">';

// Create Rating and Range ropdown

echo '

<div class="form-group required">

<label class="control-label">Rating</label>

<select name="rate\_dropdown" id="rate\_dropdown" class="form-control">';

while($row = $rateresult->fetch\_assoc()) {

$option = '<option value="'.$row["rate\_id"].'"';

if ($row["rate\_id"] == $selectedrow["rate\_id"]) {

$option .= ' selected';

}

$option .= '>'.$row["rate\_abbv"].' -- '.$row["rate\_name"].'</option><br/>';

echo $option;

}

echo '

</select>

</div>

<div class="form-group">

<label class="control-label">Range</label>

<select name="range\_dropdown" id="range\_dropdown" class="form-control">

<option value="not">No Range Set</option>';

if (!empty($selectedrow["range\_id"])) {

$ranges = array(1=>"Short-Term: 0-6 Months", 2=>"Mid-Term: 6-12 Months", 3=>"Long-Term: 12+ Months");

foreach($ranges as $rangeid=>$text) {

$option = ' <option value='.$rangeid;

if ($rangeid == $selectedrow["rate\_id"]) {

$option .= ' selected';

}

$option .= '>'.$text.'</option>';

echo $option;

}

}

else {

echo '

<option value=1>Short-Term: 0-6 Months</option>

<option value=2>Mid-Term: 6-12 Months</option>

<option value=3>Long-Term: 12+ Months</option>';

}

// Create Notes and Artifact Text Areas

echo '

</select>

</div>

<div class="form-group required">

<label class="control-label">Notes</label>

<textarea id="sysreq\_notes" class="form-control">'.$selectedrow["sysreq\_notes"].'</textarea>

</div>

<div class="form-group required">

<label class="control-label">Artifact</label>

<textarea id="artifact\_text" class="form-control">'.$selectedrow["art\_text"].'</textarea>

</div>';

}

// Requirement has not been run against system yet

else {

echo '';

}

// End "changeable\_details" div, static details div, end details row

echo '

<div class="text-center">

<button class="btn btn-primary" type="submit" disabled>Insert New Result</button>

</div>

</form>

</div>

<div id="static\_details" class="col-md-6">

<div><strong>Category: </strong>'.$reqrow["cat\_name"].'</div></br>

<div id="simple\_desc"><strong>Simple Description: </strong>'.$reqrow["req\_simple\_desc"].'</div></br>

<div id="full\_desc"><strong>Full Description: </strong>'.$reqrow["req\_desc"].'</div>

</div>

</div>';

}

?>

* search\_basic\_org(): returns the organizations of the logged in user to populate the organization dropdown menu from search\_basic.php. Is run upon report.php loading via $(document).ready(search\_basic\_org());
  + Potential errors: 'No organizations for this user. Contact your administrator for assistance.’ No organizations are found associated with the logged in user. This is a database or coding error that will require a G2 representative to correct through the backend. This error should never appear.
* search\_basic\_rest(): returns the systems of the selected organization and returns all standards in the system to their respective dropdown menus from search\_basic.php. Is run upon changing the organization dropdown and after search\_basic\_org() is successfully run.
  + Potential errors: 'No systems for this organization. Add a system to continue.' No systems are associated with the chosen system. An authorized user from the organization can follow the provided link or click “Add a System” on the main menu. See “add\_system.php” for more information.
  + 'Standards could not be retrieved. Contact your administrator for assistance.' No standards were retrieved. This is a database or coding error that will require a G2 representative to correct through the backend. This error should never appear.
* adv\_search(): returns the categories and ratings associated with the selected standard and the ranges from search\_advanced.php. This data populates the respective dropdown menus. Is run from search\_basic\_rest() and when the standard dropdown menu is changed.
  + Possible Errors: 'Could not find the requirements for the standard you chose. Contact your administrator for assistance.' No requirements were found associated with the categories associated with the chosen standard’s id. This is a database or coding error that will require a G2 representative to correct through the backend. This error should never appear.
  + 'Could not find ratings for the standard you chose. Contact your administrator for assistance.' No ratings were found associated with the root standard, the earliest version of the chosen standard. This is a database or coding error that will require a G2 representative to correct through the backend. This error should never appear.
* report(): returns the report table from create\_report.php based on all filters. Is run upon clicking the submit button. Each requirement is hyperlinked to requirement.php so that evaluation details can be added to the entry. See requirement.php for more details.
  + Possible Errors: 'Could not find requirements for conditions you set. Try widening your parameters. If you feel this is in error, contact your administrator for assistance.' No requirements returned from the standards table that have the same requirement number as system\_requirement records with the given rating and range within the given category(ies).

**PhP Execution Files:**

The mysqli group of functions coordinates data acquisition and utilization from the database:

* search\_basic.php: Runs the following queries to obtain the information necessary to populate the basic filter dropdown menus. Will return errors as referenced above if certain information is not returned.
  + SET @user = "'.$\_POST["user"].'"': Sets the @user session variable in database to the user’s id as saved in the php $\_SESSION[“user\_id”] variable. This is used as a filter by the v\_user\_reference view. See v\_user\_reference for more information.
  + SELECT DISTINCT org\_name, org\_id FROM v\_user\_reference: returns each organization the user is either working for or, if he is an MSP, a parent organization over. Will only fire if the user’s organizations have not already been previously loaded.
  + SELECT DISTINCT sys\_name, sys\_id FROM v\_user\_reference where org\_id = ('.$\_POST["org\_id"].') && sys\_id IS NOT NULL: returns all the systems associated with the chosen organization.
  + SELECT DISTINCT stand\_name, stand\_version\_rev\_num, stand\_id FROM v\_standard: returns all the standards currently in the database. Will only return the most recent versions, those with the highest stand\_version\_rev\_num. v\_standard automatically selects only those standards with the highest stand\_version\_rev\_num among each series of standards. See v\_standard for more information.
* search\_advanced.php: Runs the following queries necessary to populate the advanced filter dropdown menus. Will return errors as referenced above if certain information is not returned.
  + SET @sys = '.$\_POST["sys\_id"]: Sets the sys session variable in the database based on the sys\_id passed to the php file by adv\_search(). v\_system\_requirement results based on this variable. See v\_system\_requirement for more information.
  + SELECT DISTINCT standcat\_id, standcat\_num, cat\_name FROM v\_standard where stand\_id = '.$\_POST["stand\_id"]: gathers the category information from the standard view
  + SELECT rate\_id, rate\_abbv, rate\_name FROM rating WHERE rate\_root\_stand = '.$\_POST["stand\_id"].' OR rate\_root\_stand = (SELECT stand\_root FROM standard WHERE stand\_id = '.$\_POST["stand\_id"].')': returns the group of ratings for the selected standard by finding the root standard, the earliest version of that standard, from the chosen stand\_id.
    - Note: This presumes that rating categories will never change across a series of standards. Modifications to the database would be required if this presumption becomes inaccurate.
* The results of the two queries are compared based on the requirement ids in each. If requirements have not been ran against the specific system, the system-requirement pair will not appear in the table, thus a match will not be made against the standard array. The file will interpret this by printing out a rating of “Not Reviewed” and a range of “No Range Set.” If no range has been given for the pair, “No Range Set” will be printed for the range.
* export.php: Allows the user to export a full report of a given standard against a given system. The report will be in .csv format. Currently, custom exports cannot be created, only full reports of a single standard against a single system.
  + SELECT DISTINCT org\_nacreate\_report.php: Runs the following queries necessary to populate the report. Will return errors as referenced above if certain information is not returned.
    - SELECT DISTINCT standcat\_id, standcat\_num, cat\_name, req\_id, req\_num, req\_simple\_desc FROM v\_standard WHERE stand\_id = '.$\_POST["stand\_id"] [AND advanced conditions]: Gathers all of the requirements from the chosen standard, filtering them based on advanced filters. Default behavior of filters is to be all-inclusive.
    - SELECT sys\_id, req\_id, sysreq\_notes, rate\_id, rate\_name, range\_id, range\_desc, art\_id, art\_text FROM v\_system\_requirement WHERE '.implode(" AND ", $conditions): Gathers all the requirements that have been ran against the selected system, filtered against the advanced filters.
  + me, sys\_name, stand\_name, stand\_version\_rev\_num, standcat\_num, req\_num, art\_text, sysreq\_notes, rate\_name, range\_desc  
     FROM v\_system\_requirement sr  
     JOIN v\_user\_reference USING (sys\_id)  
     JOIN v\_standard USING (req\_id)  
     WHERE stand\_id = ".$\_GET["stand"]:  
    Gathers all the information necessary to make write to the export file. Each row will be written to export.php with the fputcsv php function. Headers as seen below will flag the information written to export.php to be downloaded as a .csv file.
* Functions.php
  + sec\_session\_start() creates a session in a secure fashion to guard against Javascript access of session variables and session hijacking. Sets security-related identifiers to cookies.
  + login() compares the given username and hash of the given password to those found in the User table. On login, sets user’s email, MySQL id, login string, and session expiration time in $\_SESSION variables. Additionally, login checks that 5 unsuccessful login attempts have not been made in the past 2 hours, else the account will be locked. An email will be sent to the user informing them of the lock and how to unlock their account. Guards against timing attacks and XSS attacks.
  + checkbrute() checks that 5 unsuccessful login attempts have not been made in the past 2 hours, else the account will be locked.
  + login\_check() checks whether the user has been idle for 15 or more minutes. If he has, then the user is logged out. This function is fired on every webpage but login.php and every execution page that is activated by clicking a link except for logout.php. If user fails this check, they are logged out and are redirected to the login page. Also checks that $\_SESSION credential variables have not been tampered with.
  + logout($loc) logs users out and returns them to the login page. Destroys their session, including all $\_SESSION variables. $loc defines the location the logout event took place to determine the correct relative path to the login page. Currently, the two possibilities are “web” for webpage files found in the root folder and “php” for execution files found in the php folder.
  + get\_orgs() retrieves the organizations a user works for or over.
  + get\_systype retrieves all system types in the database.
  + load\_requirement() creates the Requirement Detail View. Further details can be found in that page’s description.
* getData.php utilizes Google Charts API to create charts based on user data. Data is retrieved via php mysqli functions, which is then stored as a JSON object. More details can be found on the description of the Dashboard page.
* insert\_organization/system/system\_type/user.php adds the respective data point to the database through their respective add\_X.php forms.

<?php

include('php\_files\_header.php');

// Prepare the query and bind the data entered on the add user form into the procedure

$stmt = $conn->prepare('CALL insert\_systype\_sp(?,?)');

$stmt->bind\_param("ss", $tname, $u\_add);

// set parameters based on fields from add user form and execute the procedure

$tname = $\_POST["typename"];

$user = $\_SESSION['username'];

$u\_add = $user;

$stmt->execute();

// Error control

if (!$conn->errno) {

$\_SESSION["message"] = 'System-Type added successfully';

}

else if ($conn->errno == 1062) {

$\_SESSION["message"] = 'This system-type already exists.';

}

else {

$\_SESSION["message"] = 'There was an unknown database error. Contact your administrator for assistance.';

}

// Closes all statements and the DB connection

$stmt->close();

$conn->close();

header('Location: ../add\_system\_type.php');

exit;

?>

* login.php redirects the user to the home page if login() is true, else sends an error code back to the login page depending on whether the given credentials were not found in the database (error code 1), the POST variables were correctly sent (error code 2), or an attempt at unauthorized access (error code 3). Error code 0 is a successful logout. An error is shown on the login page accordingly.
* logout.php fires logout($loc), logging users out cleanly and destroying their session.

<?php

// Processes the logout request, kills sessions and deletes cookies

include('functions.php');

sec\_session\_start();

logout("php");

?>

* reset.php resets a user’s password when they have forgotten it. It creates a randomized password using bin2hex(random\_bytes(8)) creating a hexidecimal representation of 8 random binary bytes. This password is hashed before being put into the database, then is emailed to the user, letting him know how to use it to reset his permanent password.
* standard\_import.php takes a given .csv file of a standard, imports the data into the standard\_import table through the LOAD DATA LOCAL INFILE statement, then runs the standard\_import() stored procedure to insert the data into the relevant tables. See the information on the standard\_import() stored procedure for further details.
  + Possible Errors: 'This standard\'s version already exists.' This violated a unique constraint in the standard table on stand\_name and stand\_version\_rev\_num. If it is necessary to correct errors (ie. typographical), do so through DBMS directly. If absolutely necessary, and cascading drops in related tables are acceptable, drop the standard and reimport.
  + 'There was an unknown database error. Contact your administrator for assistance.' Some other error has occurred. Troubleshooting will be required. Other than formatting the .csv incorrectly, this error should never appear.
* sysreq\_import.php takes a given .csv file representing a number of requirements from any number of standards ran against any number of systems, imports them into the result\_import table with the LOAD DATA LOCAL INFILE statement, then inserts the information into the artifact and sysreq tables.

<?php

include('php\_files\_header.php');

/\* Holds the path on the user's computer and the file name of the file being imported. Also makes sure the file is CSV \*/

if (isset($\_POST['submit'])) {

$allowed = array('csv');

$filename = $\_FILES['file']['name'];

$ext = pathinfo($filename, PATHINFO\_EXTENSION);

if (!in\_array($ext, $allowed)) {

// show error message

$\_SESSION["message"] = 'Invalid file type, please use .CSV file!';

header('Location: ../sysreq\_import\_page.php');

exit;

}

else {

// If it is a CSV file, it is uploaded to the files folder on the server

move\_uploaded\_file($\_FILES["file"]["tmp\_name"], "files/" . $\_FILES['file']['name']);

$file = "files/" . $\_FILES['file']['name'];

/\* Runs the MySQL load data local infile process using the file placed in the files folder that the user uploaded \*/

// Ignore headers if user indicates they exist

if (isset($\_POST["headercheck"])) {

$stmt =

"LOAD DATA LOCAL INFILE '$file'

INTO TABLE result\_import

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 LINES

(orgname,sysname,standname,standver,catnum,reqnum,artifact,note,ratename,rangename)";

}

else {

$stmt =

"LOAD DATA LOCAL INFILE '$file'

INTO TABLE result\_import

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

(orgname,sysname,standname,standver,catnum,reqnum,artifact,note,ratename,rangename)";

}

// If unsuccessful then the user is notified

if (!mysqli\_query($conn, $stmt)) {

$\_SESSION["message"] = 'There was a problem importing your CSV. Check that it is formatted correctly.';

header('Location: ../sysreq\_import\_page.php');

exit;

}

}

}

/\* Updates the user\_name column for all new rows in the requirement import table with the user name of the person who is uploading the CSV file \*/

$sql = "UPDATE result\_import SET username=? WHERE username IS NULL";

$stmt = $conn->prepare($sql);

$stmt->bind\_param('s', $user);

$user = $\_SESSION['username'];

$stmt->execute();

$stmt->close();

/\* Runs the insert requirements procedure that inserts all new rows from the requirement import table into the main requirement table \*/

$stmt = "CALL results\_import()";

$stmt = $conn->prepare($stmt);

$stmt->execute();

// Error control

if (!$conn->errno) {

$\_SESSION["message"] = 'Report has imported successfully';

}

else if ($conn->errno == 1062) {

$\_SESSION["message"] = 'This standard\'s version already exists.';

}

else {

$\_SESSION["message"] = 'There was an unknown database error. Contact your administrator for assistance.'.$conn->error;

}

// Closes all statements and the DB connection

$stmt->close();

$conn->close();

header('Location: ../sysreq\_import\_page.php');

exit;

// Deletes the CSV file the user uploaded

unlink($file);

?>

Functions

· getuser() retrieves the value of the @user session variable set from various php execution files from the interface. To enforce access control, v\_user\_reference requires this variable for the user\_id to only show information relevant to the logged in user.

· getsys() retrieves the value of the @sys session variable set from various php execution files from the interface. To enforce access control, v\_system\_requirement requires this variable for the sys\_id to only show information relevant to the logged in user.

Procedures

· insert\_org/system/systype/user\_sp inserts data into their respective tables based on the input from their respective insert forms from the interface.

· standard\_import() transfers data from the standard\_import() table to the standard, category, standard\_category, and requirement tables. Through INSERT IGNORE statements, the procedure inserts standards, categories. standard-category pairs, and rating sets that do not already exist in the database. Through a cursor loop, each requirement in the standard\_import table is tested to see if either the full or simple description has been updated; if they have been updated, a record in the req\_amendment table is inserted to reflect these changes. Each requirement then has its stand\_cat id updated to reflect it.

· results\_import() transfers data from the result\_import table, which was imported via sysreq\_import.php, to the artifact and system\_requirement tables. Initially, it trims the trailing \r from the rangename column that exists as a result of that being the last column imported from the .csv file. Second, artifact-organization pairs that are not already present in the artifact table are added. Finally, the result information is inserted into the system\_requirement table by associating all the strings in result\_import to their respective primary keys in other tables.

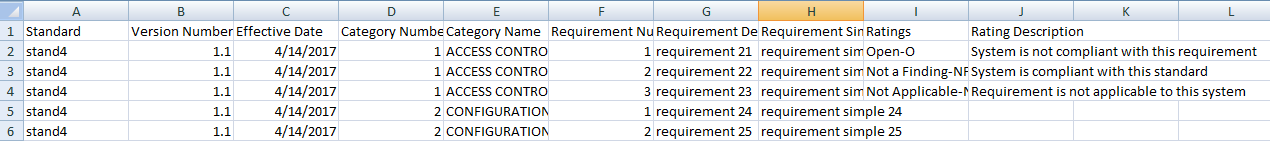
Views

· v\_user\_reference contains identifying data from the user, organization, and system table for the current user. Access control is enforced by the @user session variable which correlates to the user\_id column.

· v\_standard contains information for identifying each standard and its categories and requirements.

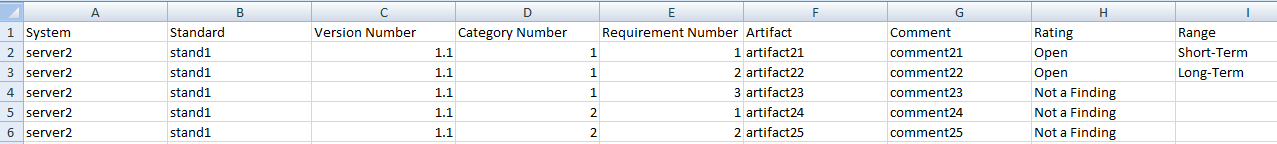
· v\_system\_requirement contains full ids and text of the results of each requirement tested against each system that has been run to date. Access control is enforced by the @sys session variable, which correlates to the sys\_id of the current system the current user has chosen to access. Selection of this system id is itself enforced by access controls within the v\_user\_reference view.

**IMPORTING**



Above is an example of the Standard Import CSV file, the following columns are needed in the order below and a short description of how the data should be entered is also provided:

* Standard – The name of the Standard given exactly how you want it in the database or how it was entered into the database at a previous time. Can be letters, numbers, or symbols and up to 100 characters long.
* Standard Version Number – the version of the standard you are entering. Can be letters, numbers, or symbols and up to 100 characters long.
* Effective date of the Standard – the date which the version of the Standard being entered with live. Has to be a date and in the format you see in the example above.
* Category Number – the number of the category being entered, can be letters, numbers, or symbols and up to 10 characters long. Enter it exactly like you want it in the database or like it was entered at a previous time.
* Category Name – The name of the Category being entered, can be letters, numbers, or symbols and up to 75 characters long. Enter it exactly like you want it in the database or like it was entered at a previous time.
* Requirement Number – The number of the Requirement being entered, can be letters, numbers, or symbols and up to 15 characters long. Enter it exactly like you want it in the database or like it was entered at a previous time.
* Requirement Description – Enter the description for the requirement, can be letters, numbers, or symbols and the amount of characters that can be given is not an issue.
* Requirement Simple Description – Enter the easier to understand description for the requirement, can be letters, numbers, or symbols and the amount of characters that can be given is not an issue.
* Ratings – Enter the rating in the given standard’s format and the abbreviation for the rating in the format you see in the example above. Both rating name and abbreviation can be letters, numbers, or symbols and up to 25 characters long for the name and 5 characters long for the abbreviation
* Rating Description – The description for the rating, letters, numbers, or symbols and up to 75 characters long.



Above is an example of the System Requirements (or Results) Import CSV file, the following columns are needed in the order below and a short description of how the data should be entered is also provided:

* System – the name of the system the result is for, has to be entered exactly like it is in the database.
* Standard – the name of the standard the result is for, has to be entered exactly like it is in the database.
* Standard Version Number – the version of the standard the result is for, has to be entered exactly like it is in the database.
* Category Number – the category number that the result falls under, has to be entered exactly like it is in the database.
* Requirement Number – the number of the requirement that the result falls under, has to be entered exactly like it is in the database.
* Artifact – the text for the organization based artifact used for the standard/category/requirement combination, has to be entered exactly like it is in the database.
* Comments/Notes for the Result – the organization can provide any notes they deem is necessary here, it can be numbers, letters, or symbols and the number of characters that can be used is very high and will not be reached.
* Rating – the name of the rating without the abbreviation, has to be entered exactly like it is in the database.
* Range – the description of the range the organization expects will be taken to satisfy the standard/category/requirement combination, has to be entered exactly like it is in the database.