LinkedOut

Requirements Analysis and System Design

Group 14

Authors

Luke Moss

Mark Madden

Scott Strothmann

Eric Song

Yaoxuan Luan

Version: 2.0

04-10-2016

Table of Contents

Website Links

Github: https://github.com/MaddenMark1495/Career-Based-Social-Network

Google Drive: https://drive.google.com/open?id=0B0t_K_ZGId4eSV9mUGIhRkVLQkE

Database Deployment

MySQL database deployed on Azure in tandem with Ubuntu Virtual Machine for website development hosting.

DDL for base database tables:

https://drive.google.com/open?id=0B_Rbp0XRbRZETFBDTzJkVnBzZmM

Link to website: http://swegroup14.centralus.cloudapp.azure.com/

Changelog

V0.1	2-25-2016	Requirement analysis rough draft created
V1.0	3-20-2016	Initial requirement analysis document completed
V1.5	3-25-2016	Sprint 1: database deployment, testing, & user interface
V2.0	4-10-2016	Spring 2: queries, stub-calls, organization UIs, user roles

Glossary

Activity Feed - Displays activity from your network, such as comments and profile changes.

Company Page - A page for companies to share information about their business and to make updates.

Connection - An indication that someone has accepted an invitation to connect, allowing the viewing of each other's profiles.

Degrees – How you are connected to others through people or groups.

Follow – A way to see updates in your news feed from people you are interested in.

Group - Allows users to come together and meet on a single page to communicate about a common subject.

Headline – Informative and concise information about your prospects.

Inbox - A place where messages can be sent, received, and viewed.

Introduction - A way for someone to bring two people who are not connected together.

Invitation - A notification to a user to join a network and form a new connection.

Mention - A tag that links a person or company in a post to their corresponding page.

Message - A private form of communication between two connected users.

Network - A group of users that one is connected to and can contact.

Profile - A page for users to share information about themselves, their job history, and experience.

Recommendation - A comment intended to commend someone, based on their experience, to viewers of their profile.

Update - Status changes and posts that appear on activity feeds.

Project Summary

The project's name is LinkedOut and it is a career-based social network designed for the professional business community. The goal of this site is to have users establish connections with other users on a professional level. A user has the option to create a profile page where they can display their professional experience and achievement, such as education and employment history. A user can then connect to other users and share this information with them.

LinkedOut has many features, but one of its main features is a messaging tool that allows users who are connected to communicate with each other privately. The application as a whole can be used in such a way that a user can create a professional brand for themselves. By doing so, a user can pursue their career passion by browsing profiles and connecting with professionals in their field, turning professional relationships into opportunities.

Functional & Non-Functional Requirements

Functional requirements:

- User can create accounts:
    - Users can set up account type
    - Users can edit account profile
- Job board:
    - Users can post jobs:
        - Users can post job descriptions/requirements
    - Users can make jobs displayed on the job board:
        - Users can make the posted jobs sorted by time or type
        - Users can save Jobs
- Apply:
    - Users can send Resume

Non-functional requirements:

- Security:
    - Users can has identification number
- Notifications:
    - Users can receive email notifications
- Search engine:
    - Users can search by university
    - Users can search by interests
    - Users can search by location
- File Review Online:
    - Users can review/edit resume online
    - Users can review/edit job descriptions
- Comments:
    - Users can post comments to company
    - Users can post comments to person

User & System Requirements

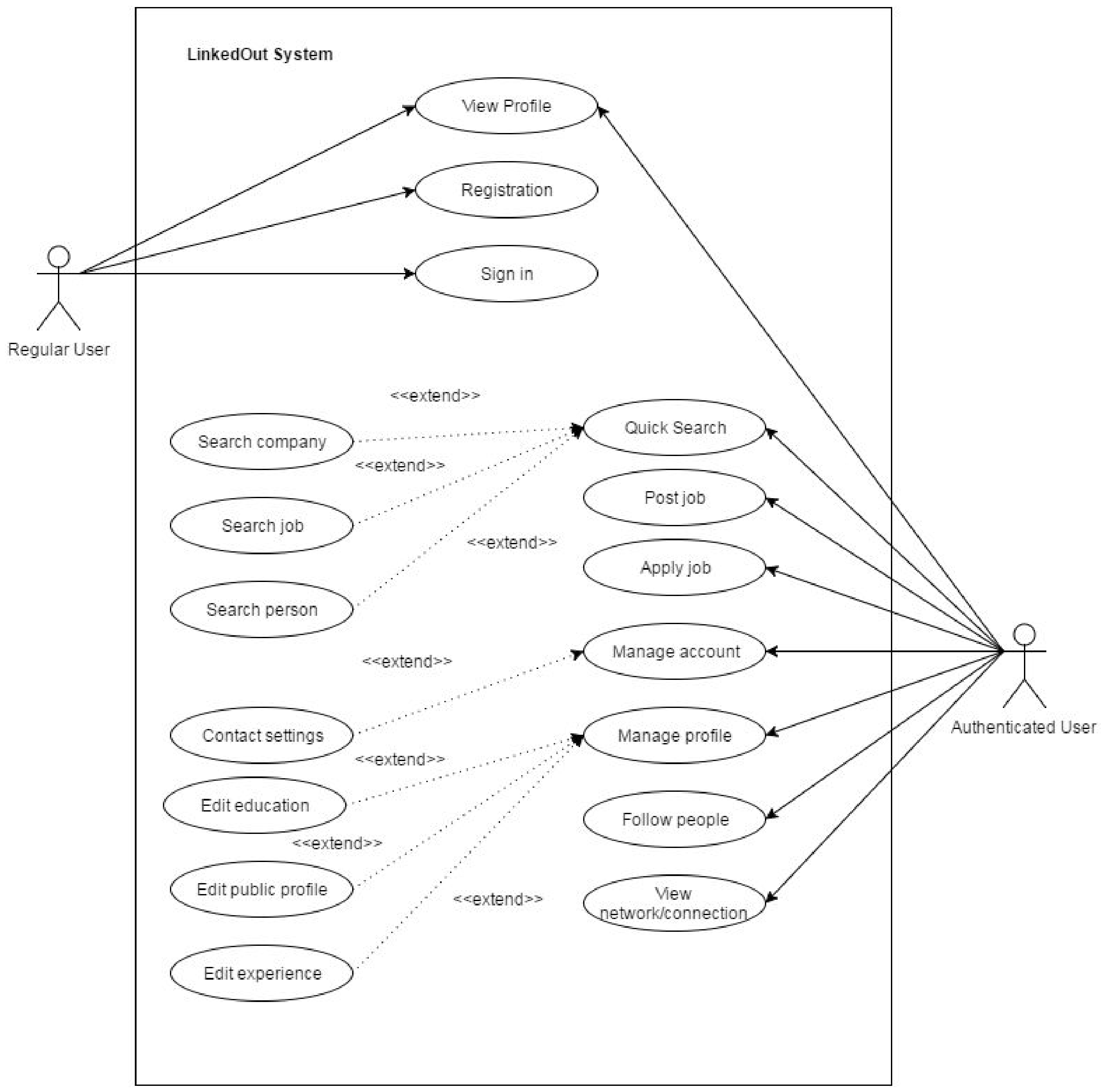
User Requirements:

- The software needs to limit what users can or cannot see based on their being logged in or not and/or linked with someone or not.
  - Use $_SESSION variables to track users' status across application usage.
  - Data - usernames, hashed passwords, links table ("friends").
- The software needs to provide registered users the ability to post resume-like work experience and education information.
  - Work experience table
  - Education table
- The software needs to allow user to search jobs.
  - The job information table
- The software needs to allow to use part of functionality without login

System Requirements:

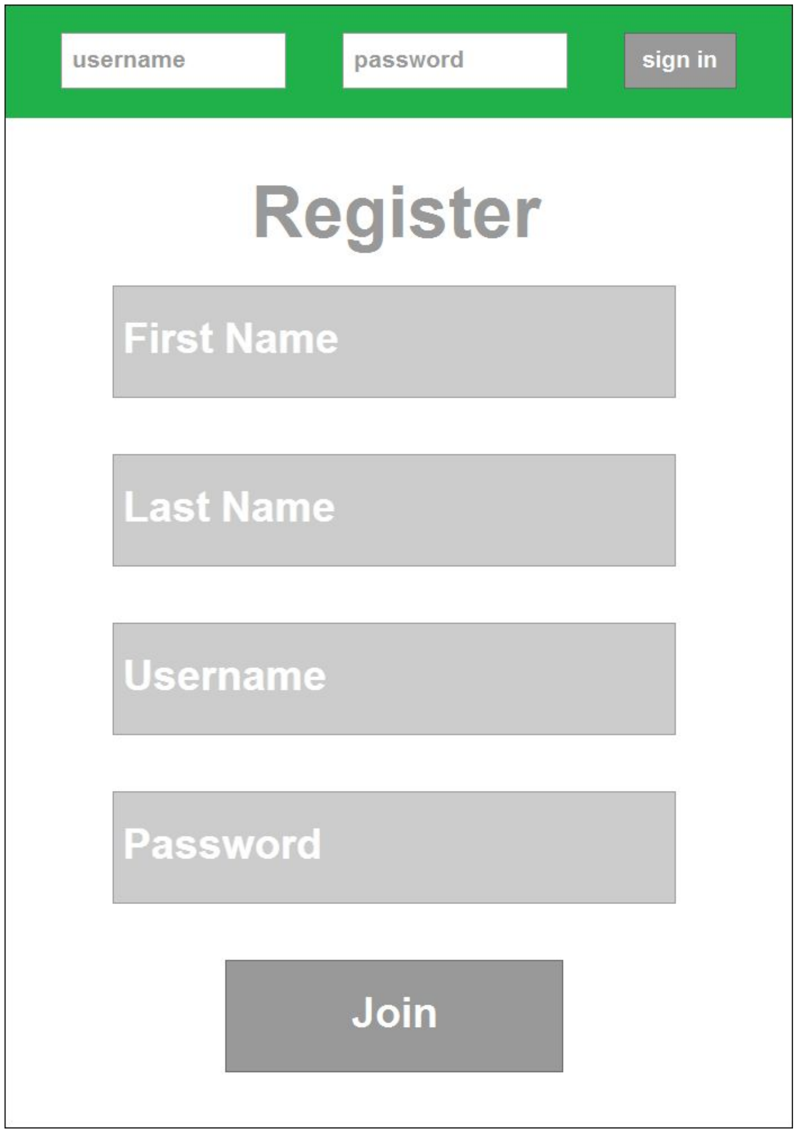
- MySQL Database
- Web server with PHP enabled
- Javascript Libraries - jQuery, D3.js
- HTML, CSS

Use Case Diagram

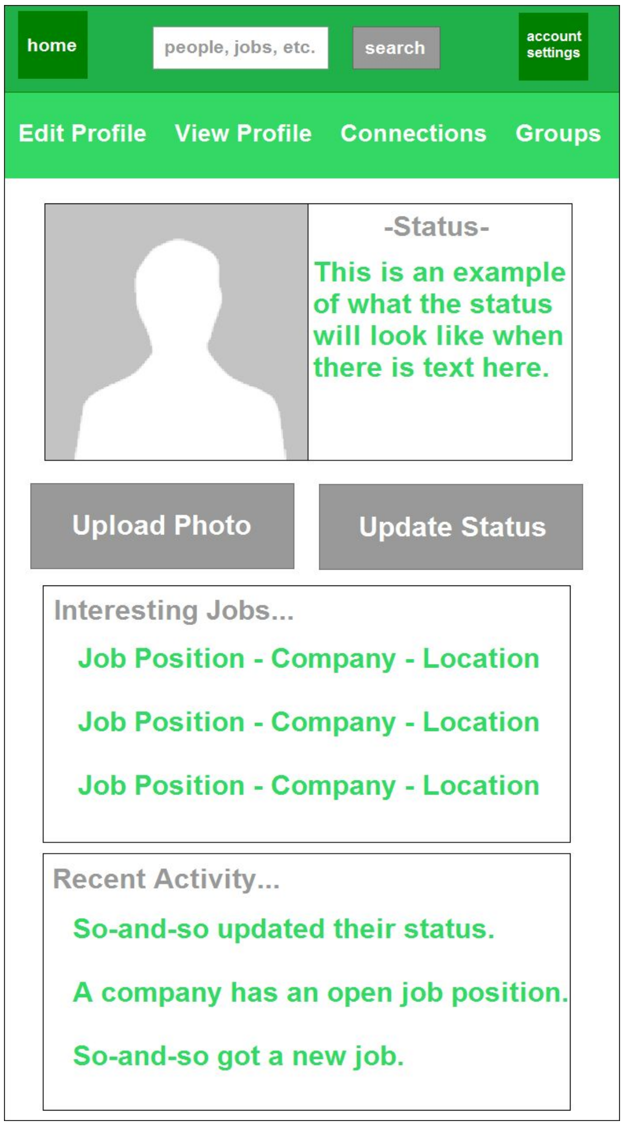


Design (User Interface)

1. Sign-in and Registering Page



2. User Home Page



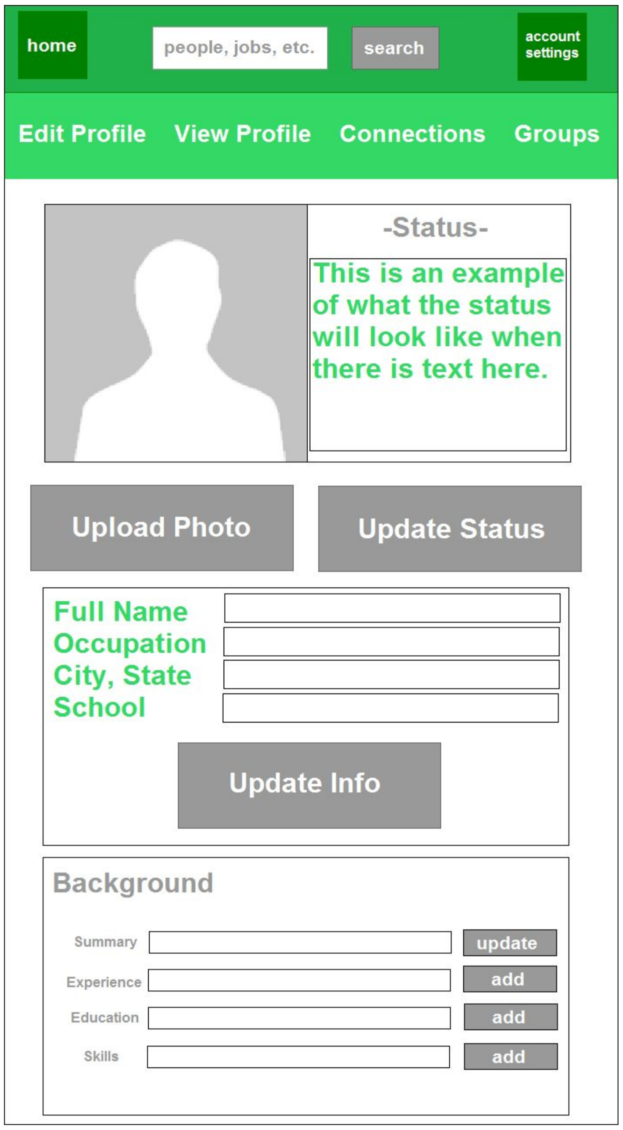
This is an example of what the status will look like when there is text here.

So-and-so updated their status.

A company has an open job position.

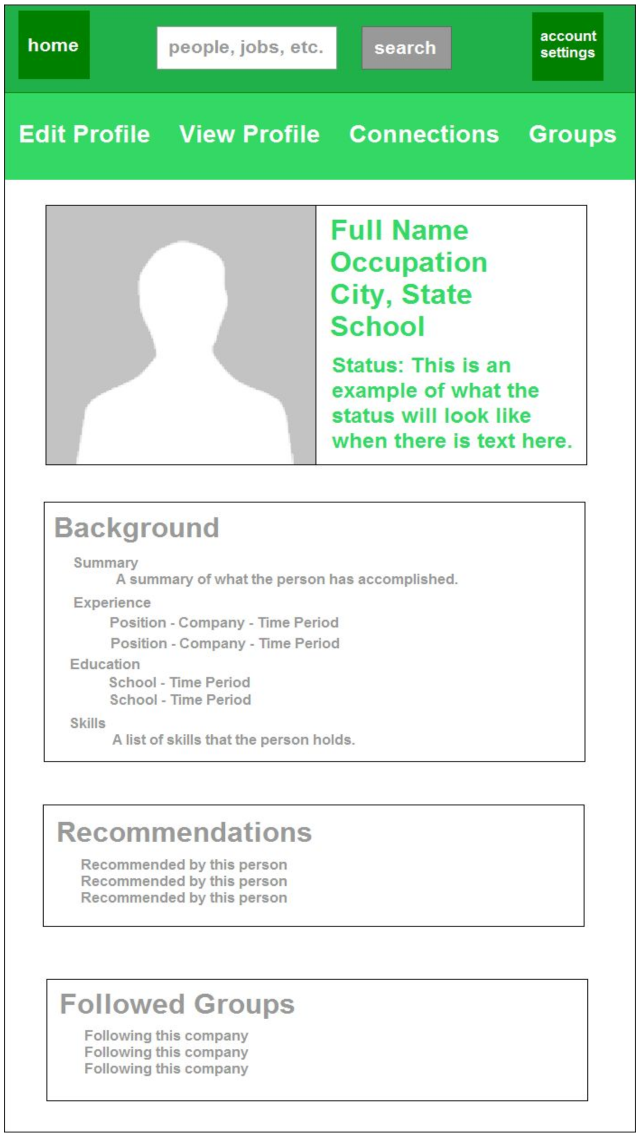
So-and-so got a new job.

3. Profile Editing Page

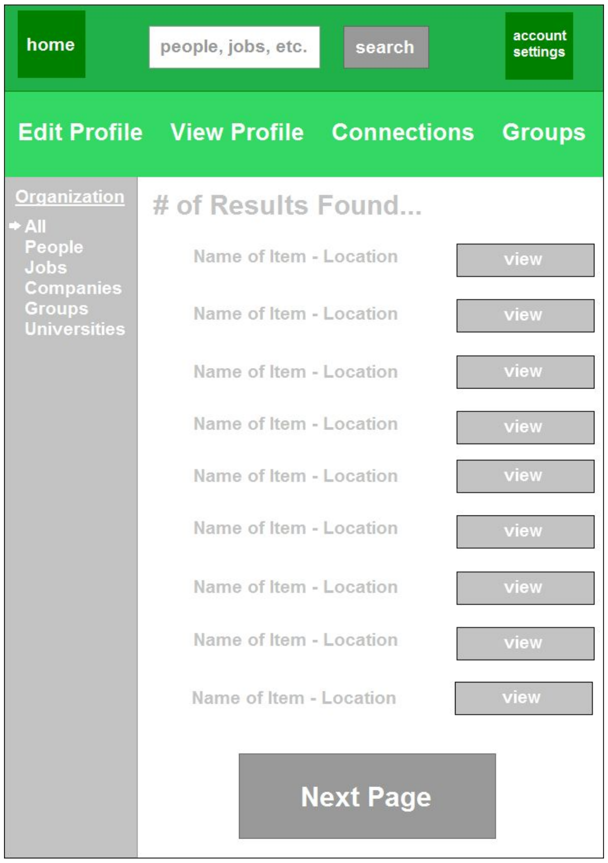


This is an example of what the status will look like when there is text here.

4. Profile View Page

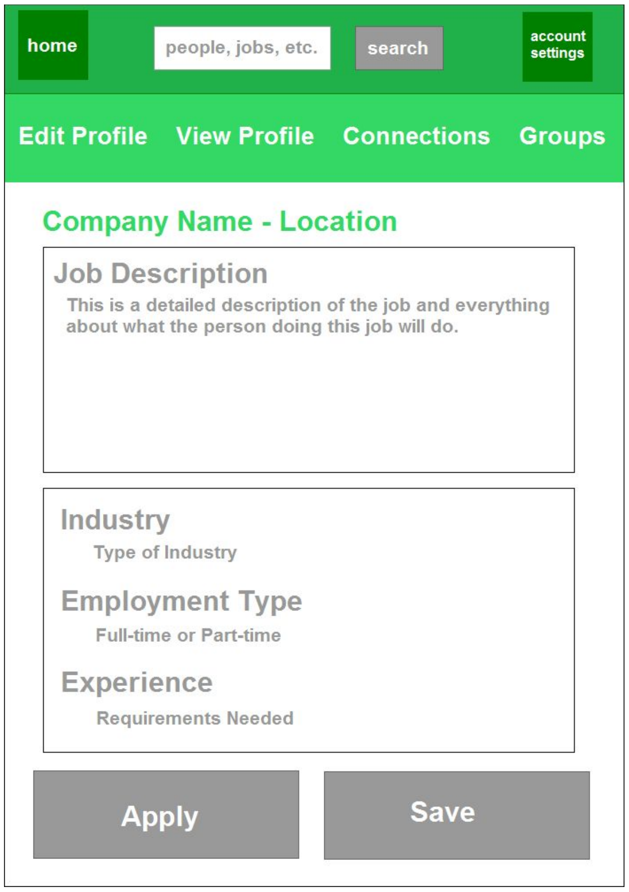

home
people, jobs, etc.
search
account settings
Edit Profile
View Profile
Connections
Groups
Full Name
Occupation
City, State
School
Status: This is an example of what the status will look like when there is text here.
Background
Summary
A summary of what the person has accomplished.
Experience
Position - Company - Time Period
Position - Company - Time Period
Education
School - Time Period
School - Time Period
Skills
A list of skills that the person holds.
Recommendations
Recommended by this person
Recommended by this person
Recommended by this person
Followed Groups
Following this company
Following this company
Following this company

5. Search Page



6. Job View Page

7. Company View Page



Company Name - Location

Job Description

This is a detailed description of the job and everything about what the person doing this job will do.

Industry

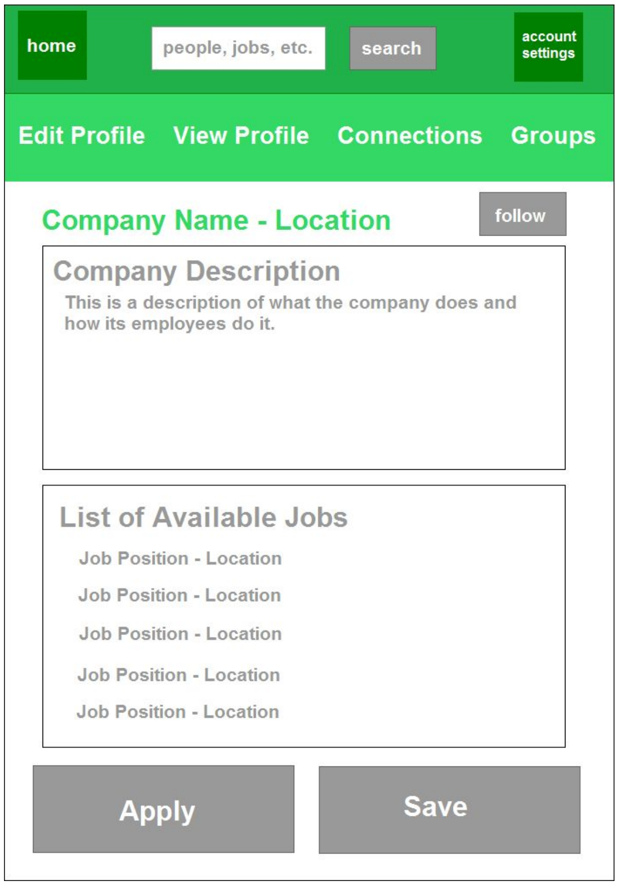
Type of Industry

Employment Type

Full-time or Part-time

Experience

Requirements Needed



Company Name - Location

Company Description

This is a description of what the company does and how its employees do it.

List of Available Jobs

Job Position - Location

Job Position - Location

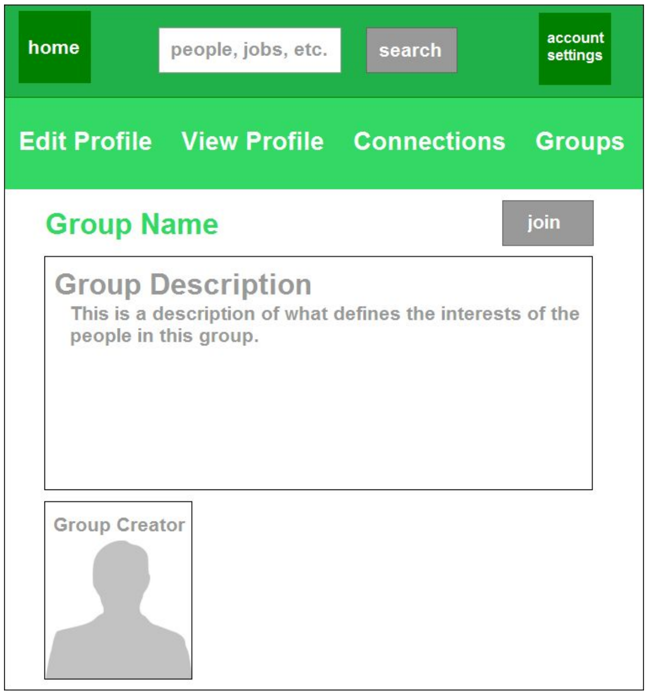
Job Position - Location

Job Position - Location

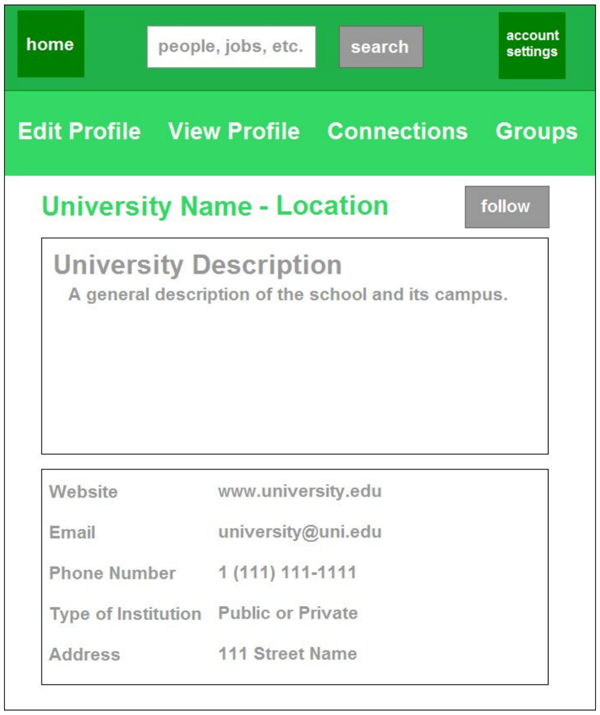
Job Position - Location

8. Group View Page

9. University View Page



Group Description
This is a description of what defines the interests of the people in this group.



University Description
A general description of the school and its campus.

Website	www.university.edu
Email	university@uni.edu
Phone Number	1 (111) 111-1111
Type of Institution	Public or Private
Address	111 Street Name

Above are the different types of screens that a user will be able to see. On the first page, which is the main page, signing in with a correct username and password by clicking the sign in button will take you to the second page. The second page is the version of your profile page that acts as your home page. To edit the profile, a user clicks the edit profile link from the nav bar to access the third page which allows users to edit all of their details. A user is then able to view the actual profile page by pressing the view profile link on the navigation bar, taking you to the fourth page. After signing into an account, a user is able to use the search bar. This takes the user to the fifth page, allowing them to see the search results. Clicking on one of the search results takes you to either a job opening page or one of the three organization pages to view all of its information.

Stub-Call Interactive UI Elements

Sign-in and Registering Page:

- Username text field for containing the username.
- Password text field for containing the password.
- Sign in button is for submitting the username and password via SQL and PHP.
- Four text fields contain first name, last name, username and password are required element for register new account.
- Clicking Join button to submit the account information to database.

Duplicate Interactive UI element:

- ❖ Since there are many duplicate buttons, I'll only describe the duplicate button one time in order to make the description clear. These button appear in all the pages below.
    - Homepage button will lead user return to the homepage.
    - Searching bar contains the search keywords.
    - User info button will display user information in database after click.
    - Add photo button will read a photo file locally.
    - Share update button will update the user information.
    - Edit profile button is switch to the profile editing page.
    - View profile button is switch to profile view page.
    - Connection button is to see the account connection.
    - Group button is to view the groups are connected to the account.

User Home Page:

- Update photo button sends update query to the database in order to update the photo.
- Update status button sends update query to the database in order to update the status information.

Profile Editing Page:

- Full name, occupation, city, states, school, summary, experience, education and skills text field contain the text will pass to the database via SQL.
- Update Info and Update button will send query order to database to update the information.

Profile View Page:

- The four text fields display the summary, background,recommendations, followed group information about this account user.

Search Page:

- Home, search,account setting, edit profile, view profile,connections and groups button which are the same to the previous page.
- The listed button in the left side of the page will display the results which only contain selected keyword.
- Next page button will display the next nine.
- View button will turn into job, company, group or university view page.

Job View Page:

- The main content in the page is the description for the job in the text area in the page which is not editable for user.
- Apply button will send request or notification to the specific user who post the job.
- Save button will save the information of employer into database for checking later.

Company View Page:

- Follow button will save the company information to the account, it can be implemented by store an ID of the company into the account database.

Group View Page:

- Text field displays the information about the group.
- Join button will send a request to the administrator or any authorized people of the group, after the administrator or the authorized people accept the request, the user will be in the group.

University View Page:

- Text field display the information such as website, contact email, phone number and address about the university.
- Follow button will save the company information to the account, it can be implemented by store an ID of the company into the account database.

Software Component (UML Diagram)

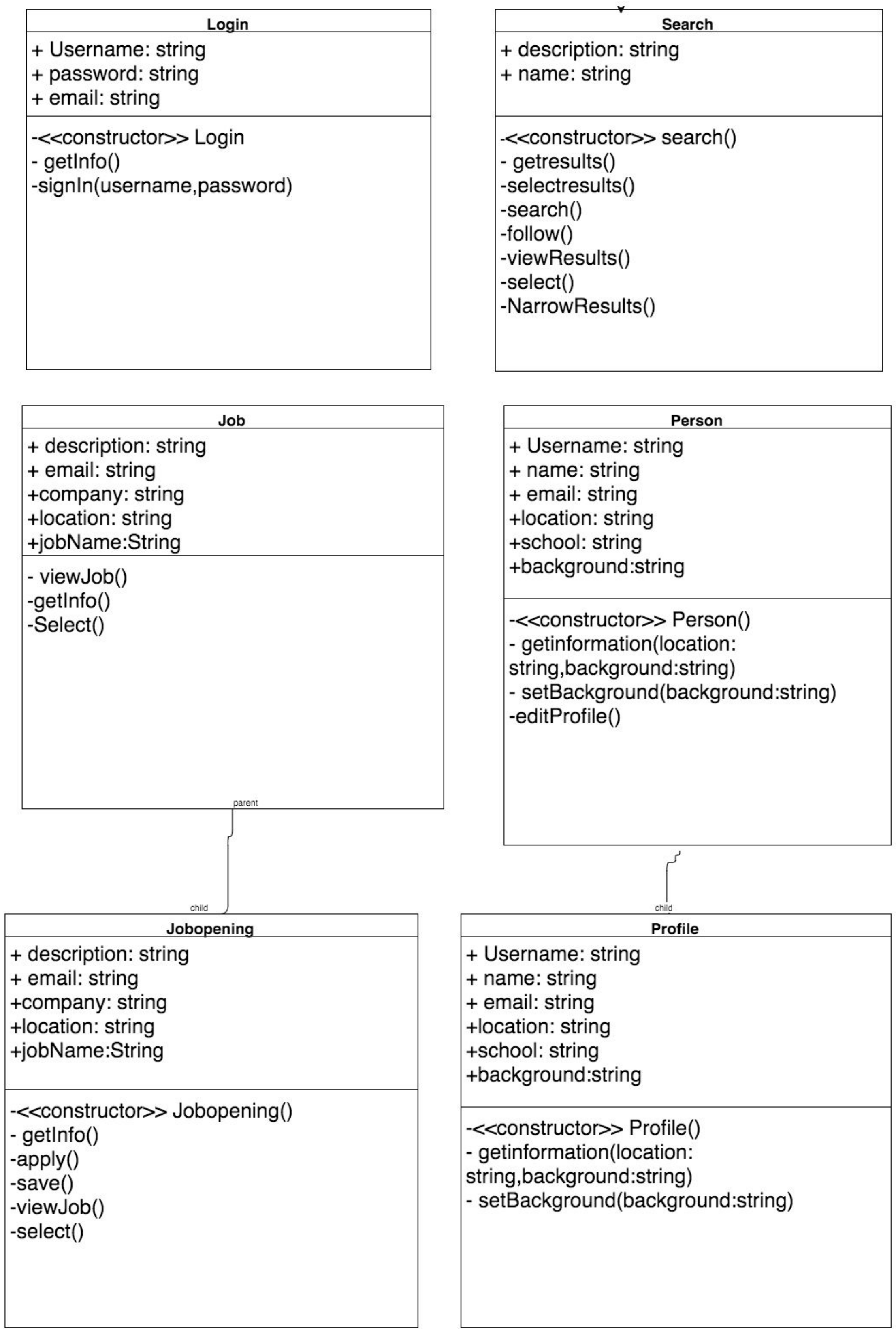
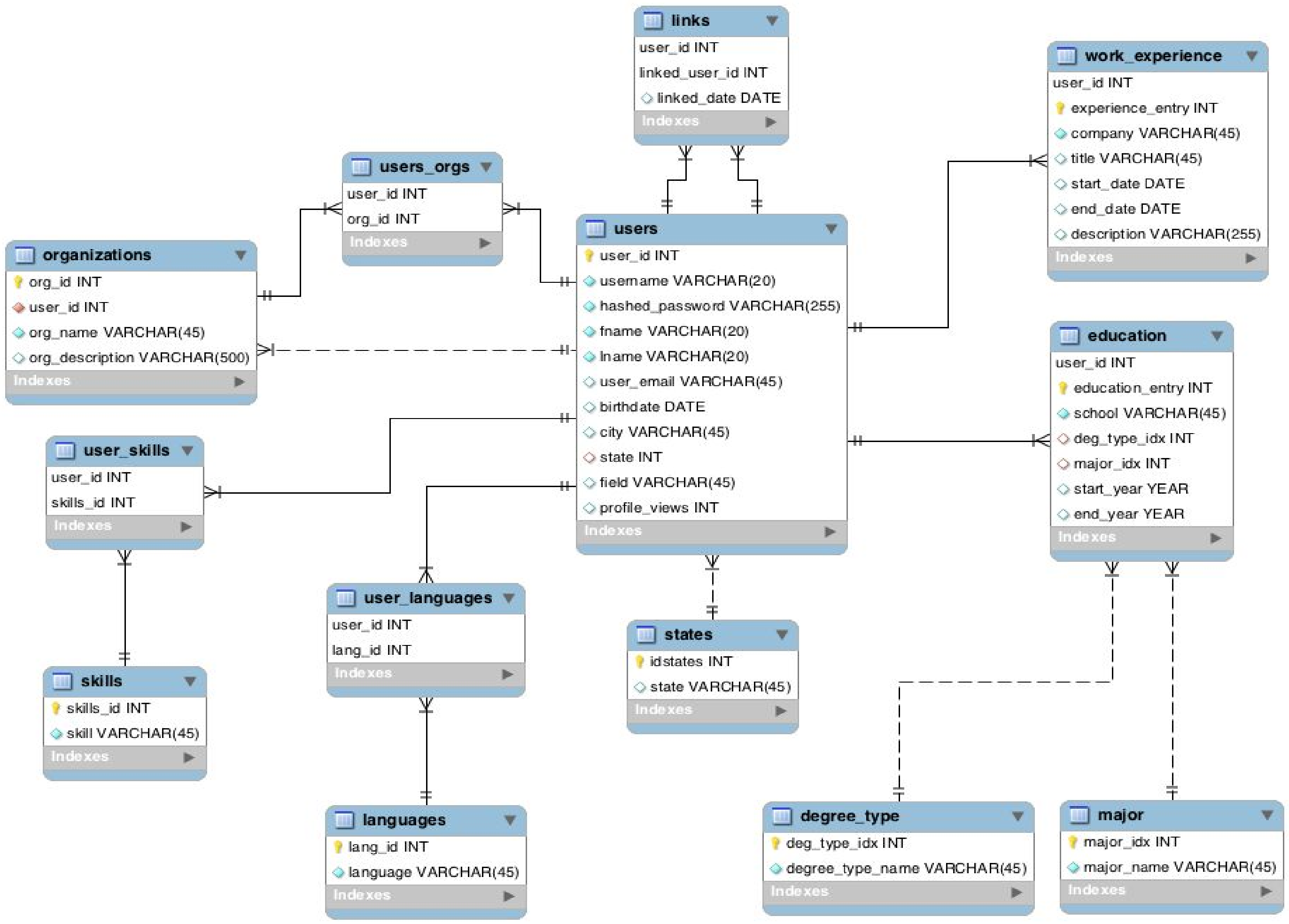


Table List (ERD)



Links table is a relation table for many to many relationship of users with other users. Lookup can go both ways (i.e. if user 1 is linked to user 2, no need to link user 2 to user 1 as a separate entry in the table). User 1 is linked to everyone where user 1 is the username and where user 1 is the linked_username.

Testing Scenarios

Activity	Testing Content	Result
Create several different accounts with different types	Account create	
Check the functionalities for different type ccount	Account type	
Add, remove content for profile	Edit profile	
Post job information with description and requirements	Job description/requirements	
Sort job board	Sort by Time/Type	
Login with correct username and password, correct username,wrong password.	User Identification	
Search keyword by the university name	University search	
Search keyword by the zip code	Location search	
Open resume online then edit it	Resume review/edit	
Make comment for person	Person comment	
Use the system without login then login the system, find out the differences	Limit user based on their being login or not Allow user use part of functionalities without login	
After login, post resume-like work experience and education information	The software needs to provide registered users the ability to post resume-like work experience and education information	
Check the work experience table and education table in the database		

Unit Testing

For Model: Database Testing

- ❖ Check the ERD of the database, find out anything incorrect with relationship.
- ❖ Insert, delete, edit data for the cell of each table in database.
- ❖ Using SQL to output the data in the database.

For Controller: PHP code

- ❖ Using php compiler to find the main problem of the code.
- ❖ Writing the test function to check each function output correctly.
- ❖ Using framework such as PHPUnit to help with unit test.
- ❖ Input data into database from website and check that it was stored properly.

Regression Testing

Regression testing is a type of software testing that verifies that software that was previously developed and tested still performs correctly after it was changed or interfaced with other software. Changes may include software enhancements, patches, configuration changes, etc. During regression testing new software bugs or *regressions* may be uncovered. Sometimes a software change impact analysis is performed to determine what areas could be affected by the proposed changes. These areas may include functional and nonfunctional areas of the system. --Wikipedia

Testing Plan:

Determining what is changed in the code and what that change means in the implementation of the view. Run the program to check whether the implementation is the same as expected. Do this regression testing every time a change is big change is added to fix bugs or make enhancement to the code.

Integration Testing

This test is to test program components that have been integrated so that users can interact with them as expected. We will use bottom up testing to perform integration testing. Bottom up testing is an approach to integrated testing where the lowest level components are tested first, then they are used to facilitate the testing of higher level components. This process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures, or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing.

We may make the integration testing after the unit testing because we need to make some components integrated. For example, PHP integrates with database, PHP integrates with HTML, various function integrates with database in HTML, functional requirements integrate with non-functional requirements.

Verification tests:	Unit Testing
	Regression testing
	Integration testing
Validation tests:	User acceptance testing

Management of users

Individuals:

Basic account

A Basic account is for anyone who wants to create and maintain professional profile online.

1. Individual users can create, edit and delete individual profiles
2. Individual users can build user professional identity on the web.
3. Individual users can build and maintain a large trusted professional network.
4. Individual users can find and reconnect with colleagues and classmates.
5. Individual users can request and provide recommendations.
6. Individual users can request up to five introductions at a time.
7. Individual users can search for and view profiles of other members.
8. Individual users can receive unlimited online messages.
9. Individual users can save up to three searches and get weekly alerts on those searches.

Premium accounts

Premium account options for job seekers, sales and talent professionals, as well as the general professional who wants to get more offline.

1. Land dream job with Job Seeker
2. Unlock sales opportunities with Sales Navigator
3. Find and hire talent with Recruiter Lite

Companies:

Only users with individual pages can create company pages

1. Company user can add and edit a company page
2. Company user can add and remove administrator for company page
3. Company user can add and remove employees to company page
4. Company user can see a list of visitors to the page
5. Company user can create, edit and delete job list of the company page

Database DDL

Database Creation SQL:
https://github.com/MaddenMark1495/Career-Based-Social-Network/blob/master/DDL/LinkedOut.sql

```
-- MySQL Script generated by MySQL Workbench

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema linkedout
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema linkedout
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `linkedout` DEFAULT CHARACTER SET utf8 ;
USE `linkedout` ;

-- -----------------------------------------------------
-- Table `linkedout`.`states`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`states` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`states` (
  `idstates` INT NOT NULL,
  `state` VARCHAR(45) NULL,
  PRIMARY KEY (`idstates`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`users`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`users` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`users` (
  `user_id` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(20) NOT NULL,
  `hashed_password` VARCHAR(255) NOT NULL,
  `fname` VARCHAR(20) NOT NULL,
  `lname` VARCHAR(20) NOT NULL,
  `user_email` VARCHAR(45) NULL,
  `birthdate` DATE NULL,
  `city` VARCHAR(45) NULL,
  `state` INT NULL,
  `field` VARCHAR(45) NULL,
  `profile_views` INT NULL DEFAULT 0,
  PRIMARY KEY (`user_id`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC),
  INDEX `state_idx` (`state` ASC),
  UNIQUE INDEX `user_id_UNIQUE` (`user_id` ASC),
  CONSTRAINT `user_state`
        FOREIGN KEY (`state`)
        REFERENCES `linkedout`.`states` (`idstates`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`work_experience`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`work_experience` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`work_experience` (
  `user_id` INT NOT NULL,
  `experience_entry` INT NOT NULL,
  `company` VARCHAR(45) NOT NULL,
  `title` VARCHAR(45) NULL,
  `start_date` DATE NULL,
  `end_date` DATE NULL,
  `description` VARCHAR(255) NULL,
  PRIMARY KEY (`user_id`, `experience_entry`),
  CONSTRAINT `w_user`
        FOREIGN KEY (`user_id`)
        REFERENCES `linkedout`.`users` (`user_id`)
        ON DELETE CASCADE
        ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `linkedout`.`degree_type`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`degree_type` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`degree_type` (
  `deg_type_idx` INT NOT NULL AUTO_INCREMENT,
  `degree_type_name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`deg_type_idx`),
  UNIQUE INDEX `deg_type_idx_UNIQUE` (`deg_type_idx` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`major`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`major` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`major` (
  `major_idx` INT NOT NULL AUTO_INCREMENT,
  `major_name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`major_idx`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`education`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`education` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`education` (
  `user_id` INT NOT NULL,
  `education_entry` INT NOT NULL,
  `school` VARCHAR(45) NOT NULL,
  `deg_type_idx` INT NULL,
  `major_idx` INT NULL,
  `start_year` YEAR NULL,
  `end_year` YEAR NULL,
  PRIMARY KEY (`user_id`, `education_entry`),
  INDEX `deg_type_idx_idx` (`deg_type_idx` ASC),
  INDEX `major_idx_idx` (`major_idx` ASC),
  CONSTRAINT `ed_user`
```

```
          FOREIGN KEY (`user_id`)
          REFERENCES `linkedout`.`users` (`user_id`)
          ON DELETE CASCADE
          ON UPDATE NO ACTION,
  CONSTRAINT `deg_type_idx`
          FOREIGN KEY (`deg_type_idx`)
          REFERENCES `linkedout`.`degree_type` (`deg_type_idx`)
          ON DELETE NO ACTION
          ON UPDATE NO ACTION,
  CONSTRAINT `major_idx`
          FOREIGN KEY (`major_idx`)
          REFERENCES `linkedout`.`major` (`major_idx`)
          ON DELETE NO ACTION
          ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`links`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`links` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`links` (
  `user_id` INT NOT NULL,
  `linked_user_id` INT NOT NULL,
  `linked_date` DATE NULL,
  PRIMARY KEY (`user_id`, `linked_user_id`),
  INDEX `linked_username_idx` (`linked_user_id` ASC),
  CONSTRAINT `user1`
          FOREIGN KEY (`user_id`)
          REFERENCES `linkedout`.`users` (`user_id`)
          ON DELETE CASCADE
          ON UPDATE NO ACTION,
  CONSTRAINT `user2`
          FOREIGN KEY (`linked_user_id`)
          REFERENCES `linkedout`.`users` (`user_id`)
          ON DELETE CASCADE
          ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`organizations`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`organizations` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`organizations` (
  `org_id` INT NOT NULL,
  `user_id` INT NOT NULL,
  `org_name` VARCHAR(45) NOT NULL,
  `org_description` VARCHAR(500) NULL,
  PRIMARY KEY (`org_id`),
  INDEX `org_admin_idx` (`user_id` ASC),
  CONSTRAINT `org_admin`
	FOREIGN KEY (`user_id`)
	REFERENCES `linkedout`.`users` (`user_id`)
	ON DELETE NO ACTION
	ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`users_orgs`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`users_orgs` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`users_orgs` (
  `user_id` INT NOT NULL,
  `org_id` INT NOT NULL,
  PRIMARY KEY (`user_id`, `org_id`),
  INDEX `org_idx` (`org_id` ASC),
  CONSTRAINT `uo_user`
	FOREIGN KEY (`user_id`)
	REFERENCES `linkedout`.`users` (`user_id`)
	ON DELETE CASCADE
	ON UPDATE NO ACTION,
  CONSTRAINT `uo_org`
	FOREIGN KEY (`org_id`)
	REFERENCES `linkedout`.`organizations` (`org_id`)
	ON DELETE NO ACTION
	ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`skills`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`skills` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`skills` (
  `skills_id` INT NOT NULL AUTO_INCREMENT,
  `skill` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`skills_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`user_skills`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`user_skills` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`user_skills` (
  `user_id` INT NOT NULL,
  `skills_id` INT NOT NULL,
  PRIMARY KEY (`user_id`, `skills_id`),
  INDEX `us_skill_idx` (`skills_id` ASC),
  CONSTRAINT `us_user`
          FOREIGN KEY (`user_id`)
          REFERENCES `linkedout`.`users` (`user_id`)
          ON DELETE CASCADE
          ON UPDATE NO ACTION,
  CONSTRAINT `us_skill`
          FOREIGN KEY (`skills_id`)
          REFERENCES `linkedout`.`skills` (`skills_id`)
          ON DELETE NO ACTION
          ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `linkedout`.`languages`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`languages` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`languages` (
  `lang_id` INT NOT NULL AUTO_INCREMENT,
  `language` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`lang_id`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `linkedout`.`user_languages`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `linkedout`.`user_languages` ;

CREATE TABLE IF NOT EXISTS `linkedout`.`user_languages` (
  `user_id` INT NOT NULL,
  `lang_id` INT NOT NULL,
  PRIMARY KEY (`user_id`, `lang_id`),
  INDEX `ul_lang_idx` (`lang_id` ASC),
  CONSTRAINT `ul_user`
        FOREIGN KEY (`user_id`)
        REFERENCES `linkedout`.`users` (`user_id`)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
  CONSTRAINT `ul_lang`
        FOREIGN KEY (`lang_id`)
        REFERENCES `linkedout`.`languages` (`lang_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Insert Statements:

```
--new user registration
INSERT INTO `linkedout`.`users` (`username`, `hashed_password`, `fname`, `lname`)
   VALUES (?, ?, ?, ?);

--link request accepted
INSERT INTO `linkedout`.`links` (`user_id`, `linked_user_id`, `linked_date`)
   VALUES (?, ?, CURDATE());

--add new major
INSERT INTO `linkedout`.`major` (`major_name`)
   VALUES (?);

--add new degree type
```

```
INSERT INTO `linkedout`.`degree_type` (`degree_type`)
   VALUES (?);

--add education entry to resume
INSERT INTO `linkedout`.`education` (`user_id`, `education_entry`, `school`,
`deg_type_idx`, `major_idx`, `start_year`, `end_year`)
   VALUES (?, ?, ?, ?, ?, ?, ?);

--insert work experience entry
INSERT INTO `linkedout`.`work_experience` (`user_id`, `experience_entry`, `company`,
`title`, `start_date`, `end_date`, `description`)
   VALUES (?, ?, ?, ?, ?, ?, ?);

--add new skill
INSERT INTO `linkedout`.`skills` (`skill`)
   VALUES (?);

--add user skill
INSERT INTO `linkedout`.`user_skills` (`user_id`,`skill_id`)
   VALUES (?, (SELECT `skill_id` FROM `linkedout`.`skills` WHERE `skill_name` = ?));

--add new language
INSERT INTO `linkedout`.`languages` (`language`) VALUES (?);

--add user language
INSERT INTO `linkedout`.`user_languages` (`user_id`, `lang_id`)
   VALUES (?, (SELECT `lang_id` FROM `linkedout`.`languages` WHERE `language` =
?));

--add new organization
INSERT INTO `linkedout`.`organizations` (`user_id`, `org_name`, `desription`)
   VALUES (?, ?, ?);

--add new organization member
INSERT INTO `linkdedout`.`user_orgs` (`user_id`, `org_id`)
   VALUES (?, ?);
```

Update Statements:

```
--general profile information change
UPDATE `linkedout`.`users`
   SET /*col1=val1, col2=val2, etc*/
   WHERE `user_id` = ?;

--update education entry
UPDATE `linkedout`.`education`
   SET /*col1=val1, col2=val2, etc*/
   WHERE `user_id` = ? AND `education_entry` = ?;

--update work_experience entry
UPDATE `linkedout`.`work_experience`
   SET /*col1=val1, col2=val2, etc*/
   WHERE `user_id` = ? AND `education_entry` = ?;

--update organization
UPDATE `linkedout`.`organizations`
   SET /*col1=val1, col2=val2, etc*/
   WHERE `org_id` = ?;
```

Delete Statements:

```
--delete user profile (cascade deletes all other associations with this user_id)
DELETE FROM `linkedout`.`users`
   WHERE `user_id` = ?;

--delete user skill
DELETE FROM `linkedout`.`user_skills`
   WHERE `user_id` = ?;

--delete user language
DELETE FROM `linkedout`.`user_languages`
   WHERE `user_id` = ?;

--delete user organization
DELETE FROM `linkedout`.`user_orgs`
   WHERE (`user_id` = ? AND `org_id` = ?);

--delete linked friend (not guaranteed which column user_ids are in so have to check for
both directions)
DELETE FROM `linkedout`.`links`
   WHERE (`user_id` = ? AND `linked_user_id` = ?)
         OR (`user_id` = ? AND `linked_user_id` = ?);

--delete education entry
DELETE FROM `linkedout`.`education`
   WHERE (`user_id` = ? AND `education_entry` = ?);

--delete work experience entry
DELETE FROM `linkedout`.`work_experience`
   WHERE (`user_id` = ? AND `experience_entry` = ?);
```