Final Submission

Group 14

Authors

Luke Moss

Mark Madden

Scott Strothmann

Eric Song

Yaoxuan Luan

Version: 4.0

05-09-2016

Table of Contents

- ❑ Links 2
- ❑ Changelog 3
- ❑ Glossary 4
- ❑ Project Summary 5
- ❑ Functional & Non-Functional Requirements 6
- ❑ User & System Requirements 7
- ❑ Use Case Diagram 8
- ❑ User Interface Design 9
- ❑ Stub-calls for Interactive UI Element 14
- ❑ Software Component (UML Diagram) 16
- ❑ ERD 17
- ❑ Testing
  - ➢ Testing Scenarios 18
  - ➢ Unit Testing 19
  - ➢ Regression Testing 19
  - ➢ Integration Testing 20
- ❑ Management of Users 22
- ❑ Database DDL 23
- ❑ Sprint 3: Development 39
- ❑ Data Visualization 40
- ❑ Edge Cases 41
- ❑ Sprint 4: Development 42
- ❑ Sprint 4: Issues 42
- ❑ Next Steps 43

Links

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

Website:

http://swegroup14.centralus.cloudapp.azure.com/

Please report any issues through GitHub, and thanks for checking out our application.

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	Sprint 2: queries, stub-calls, organization UIs, user roles
V3.0	4-18-2016	Sprint 3: PHP scripting, HTML UI template, Data visualization, Edge cases
V3.5	4-25-2016	Sprint 4: Updated website links editprofile.php - mostly functional - WIP Top10.php Data Visualization - Done connections.php - mostly functional - WIP Updated ERD and Database Creation SQL script Sprint 4: Development and Issues
V4.0	5-9-2016	Final Submission: Data Visualization Screenshots (per Sprint 4) Sprint 4: Issues - SOLVED Next Steps

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

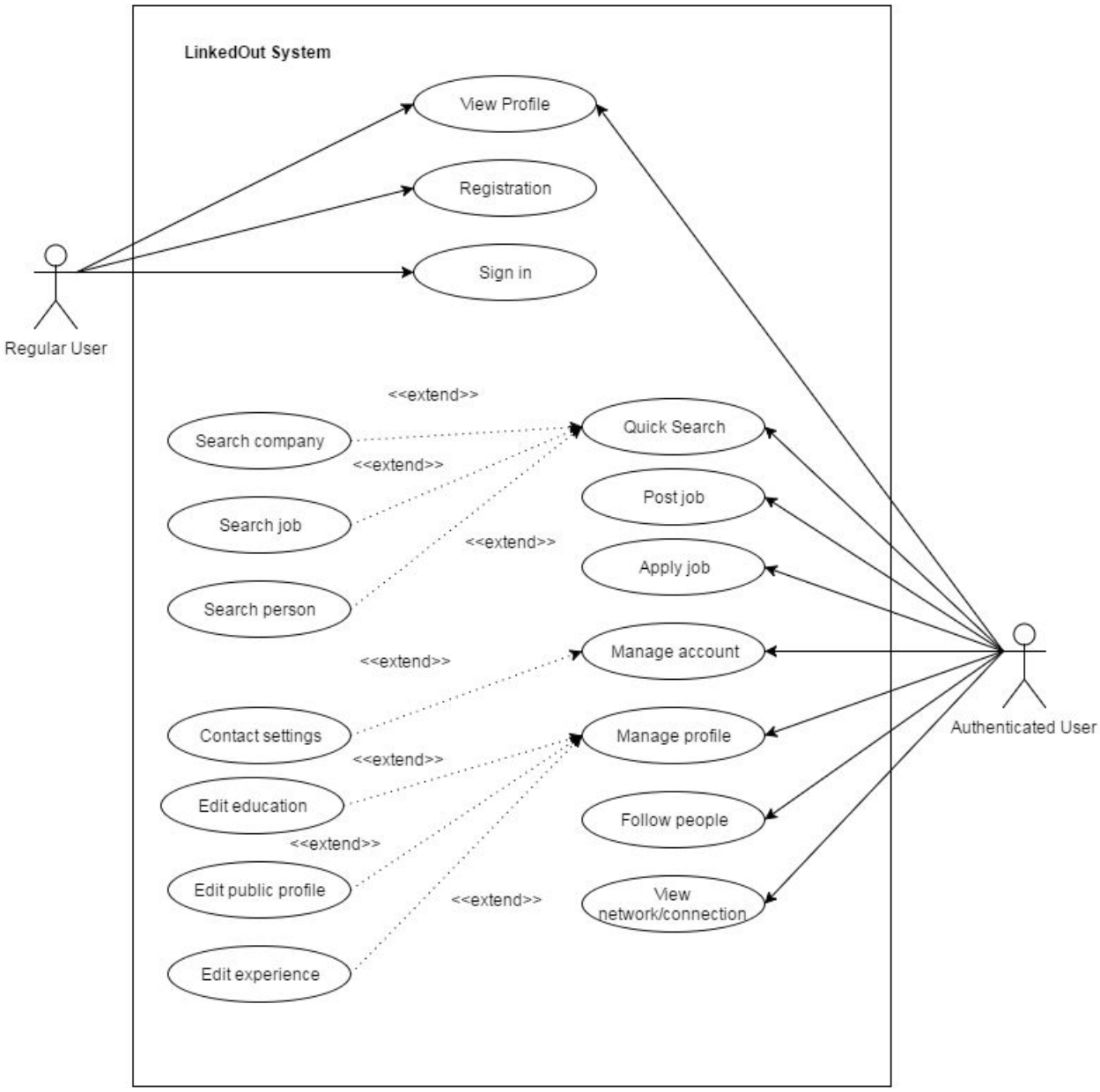
User Requirements:

- 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:

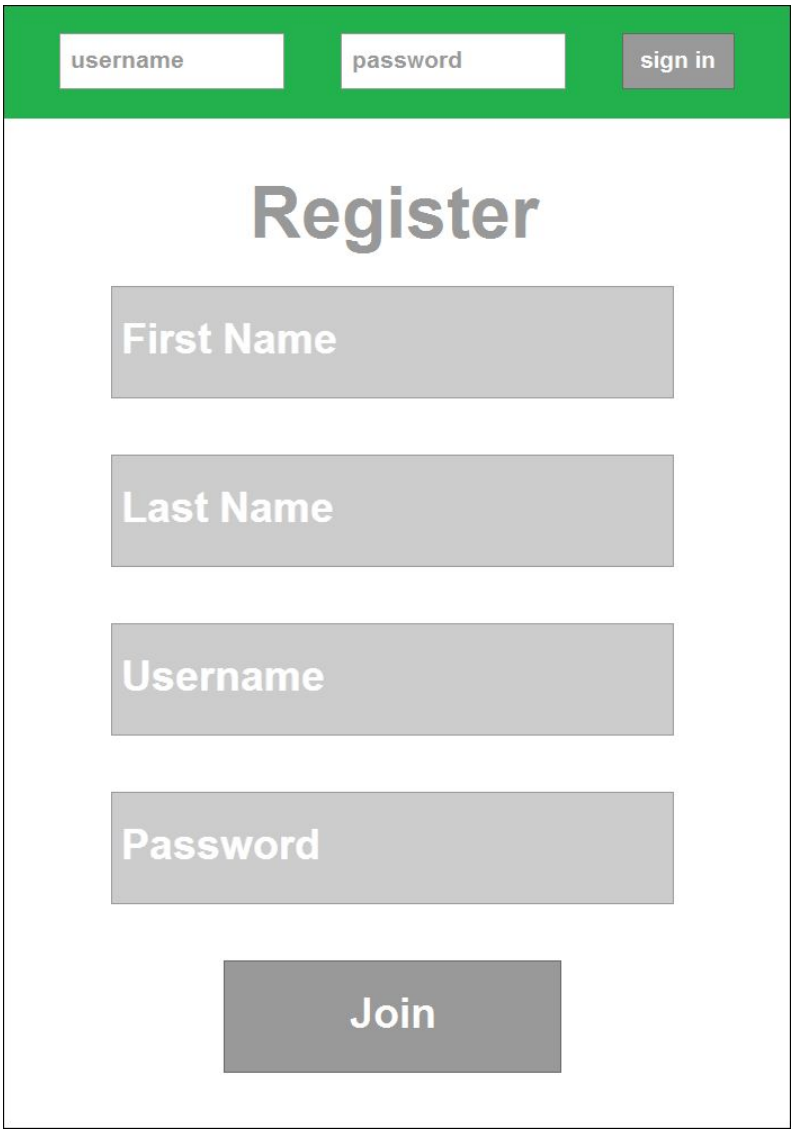
- MySQL Database
- Web server with PHP enabled
- Javascript Libraries - jQuery, D3.js
- HTML, CSS

Use Case Diagram

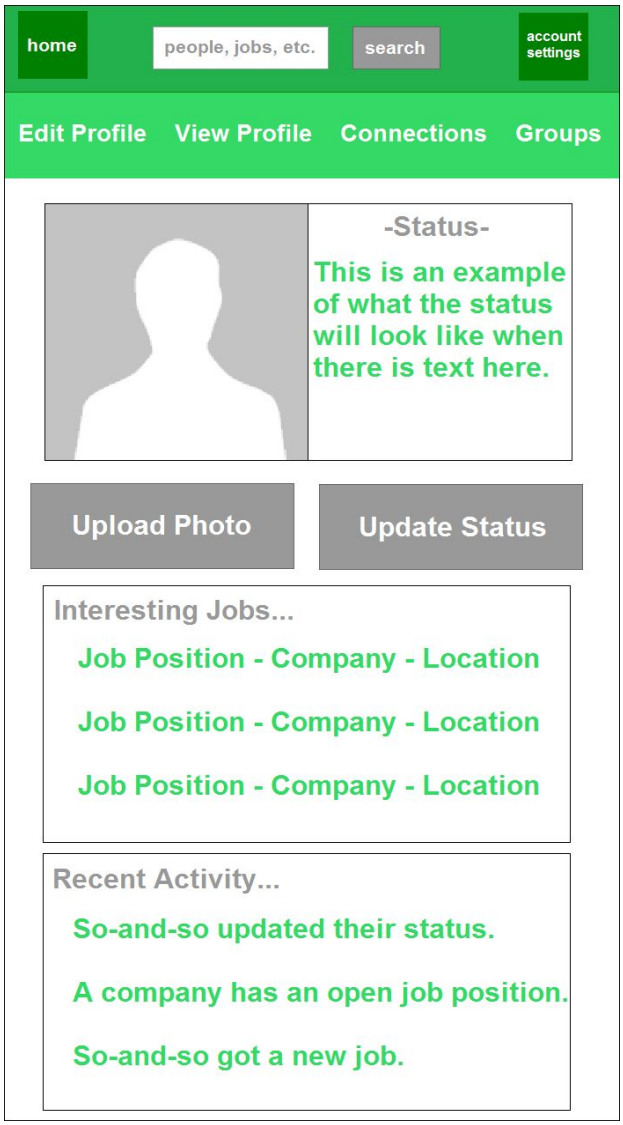


Design (User Interface)

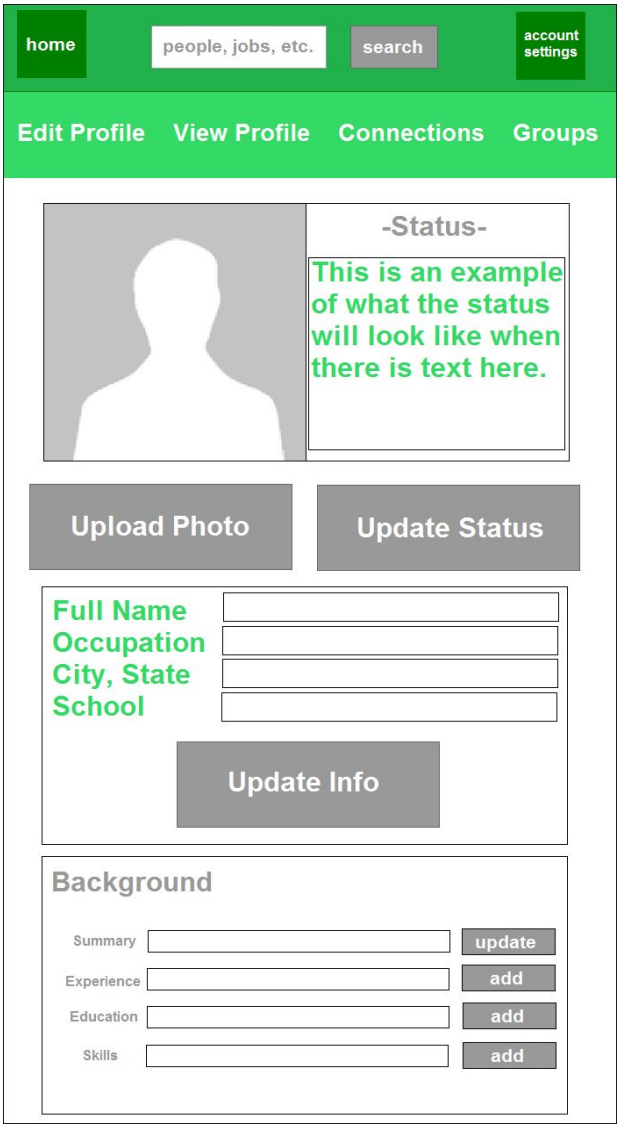
1. Sign-in and Registering Page



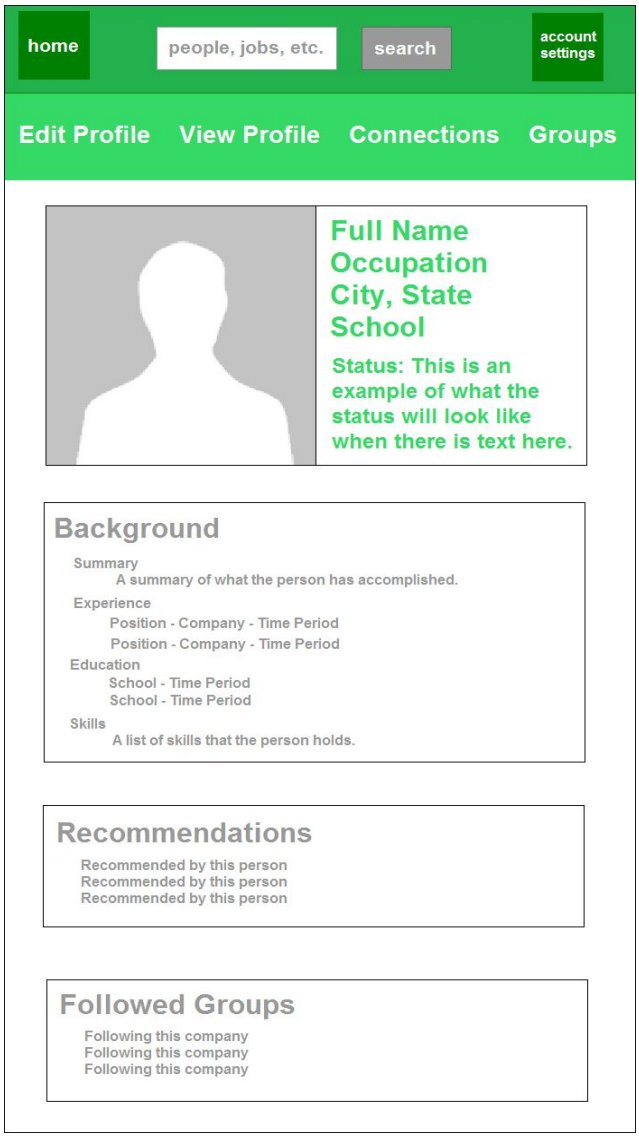
2. User Home Page



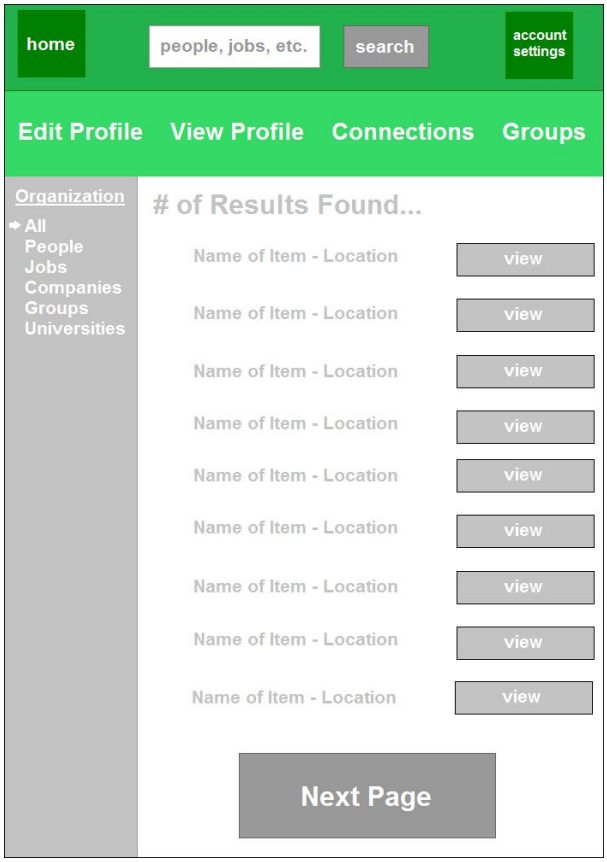
3. Profile Editing Page



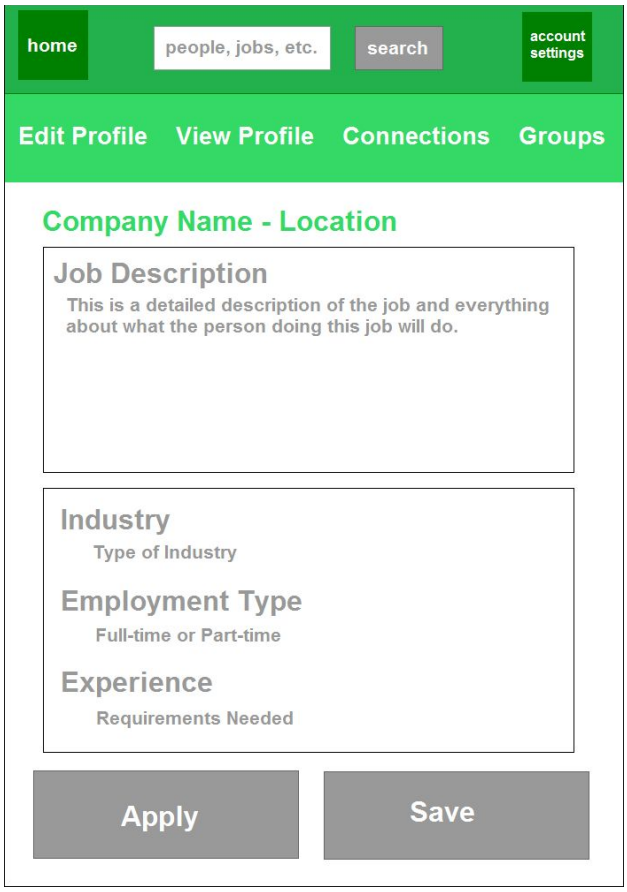
4. Profile View Page



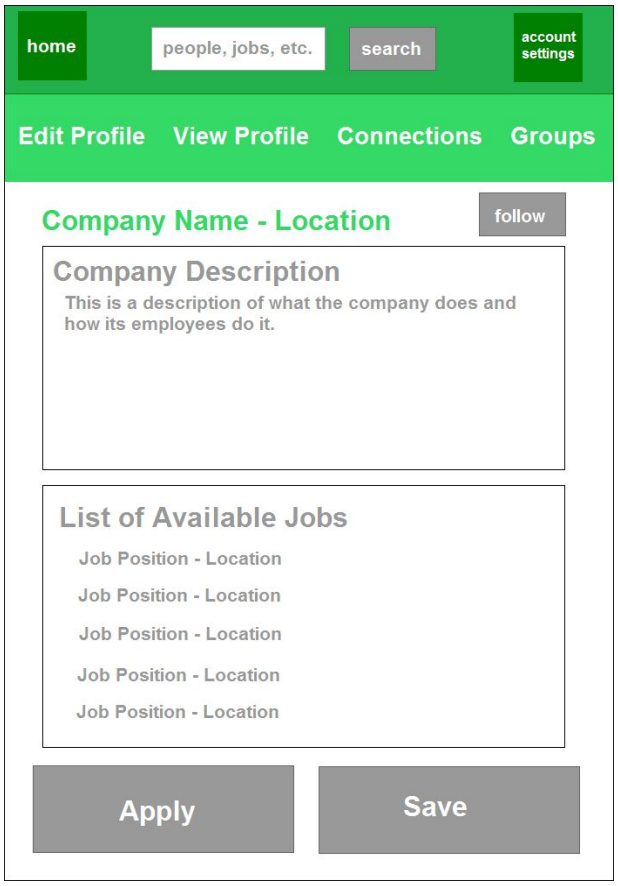
5. Search Page



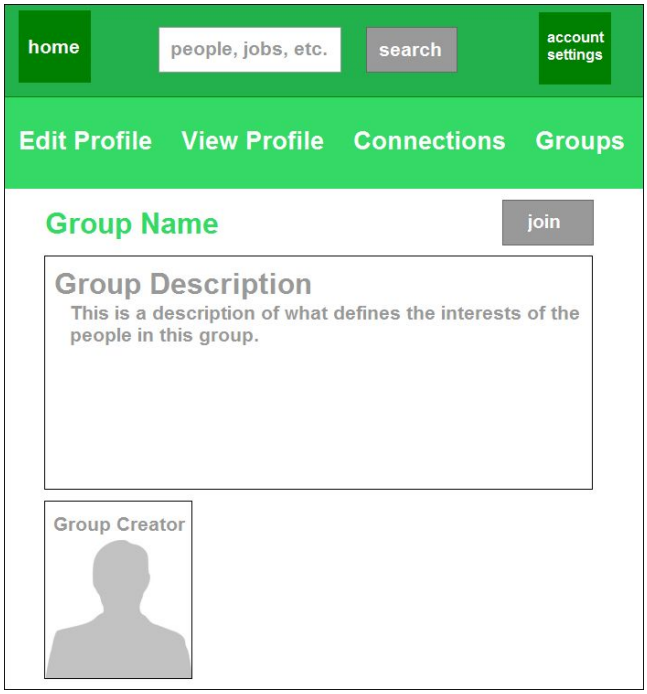
6. Job View Page



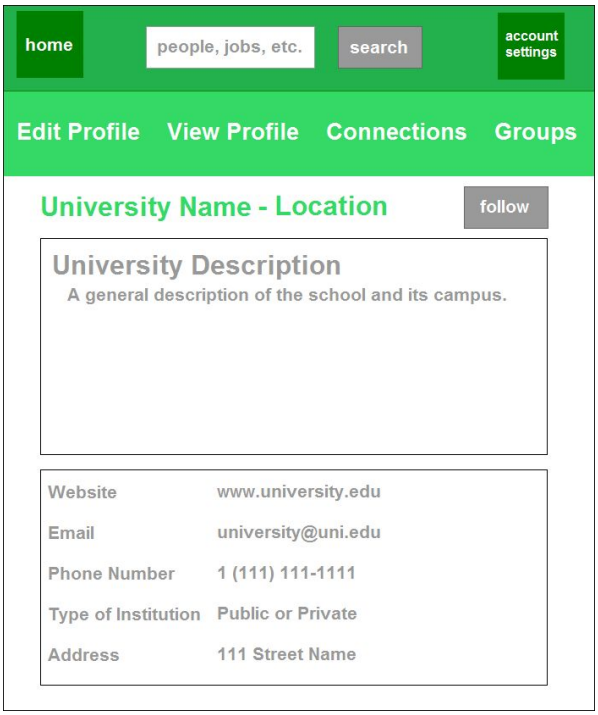
7. Company View Page



8. Group View Page



9. University View Page



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-calls for Interactive UI element

Sign-in and Registering Page:

- Sign in button is for submitting the username and password via SQL and PHP.
    - <input type="text" name="name" id="accountName">
      Function function($username,$password)
      Return 1 when the correct password and username, otherwise return 0;
- Clicking Join button to submit the account information to database.
    - Function createuser($username,$password)
      No return

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.
        - Home
    - Searching bar contains the search keywords.
        - Function search($keyword)
          Return search result
    - Account setting
        - Account setting
    - Edit profile button is switch to the profile editing page.
        - Editing Profile
    - View profile button is switch to profile view page.
        - View Profile
    - Connection button is to see the account connection.
        - Connection
    - Group button is to view the groups are connected to the account.
        - Group

User Home Page:

- Update photo button sends update query to the database in order to update the photo.
    - Function updatePic($pic)
- Update status button sends update query to the database in order to update the status information.
    - Function updateStatus($status,$userid)

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.

    - Function updateInfo($info,$userid)

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.
    - Function List($keyword)
      Return $result which contain adapt to the keyword
- Next page button will display the next nine.
- View button will turn into job, company, group or university view page.
    - Function List($JobId)
      Return the column about specific job information fit the job id

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.
    - Function applyMess($accountinfo,$userid)
      Return 1 when the message send successfully, otherwise return 0
- Save button will save the information of employer into database for checking later, actually is update progress in DB
    - Function save($Info,$userid)

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.
    - Function follow($company,$userid)
      Return 1 when query works correctly otherwise return 0

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.
    - Function request($groupid,$userid)
      Return 1 when send message correctly otherwise return 1 display error message

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.
    - Function follow($company)

Software Component (UML Diagram)

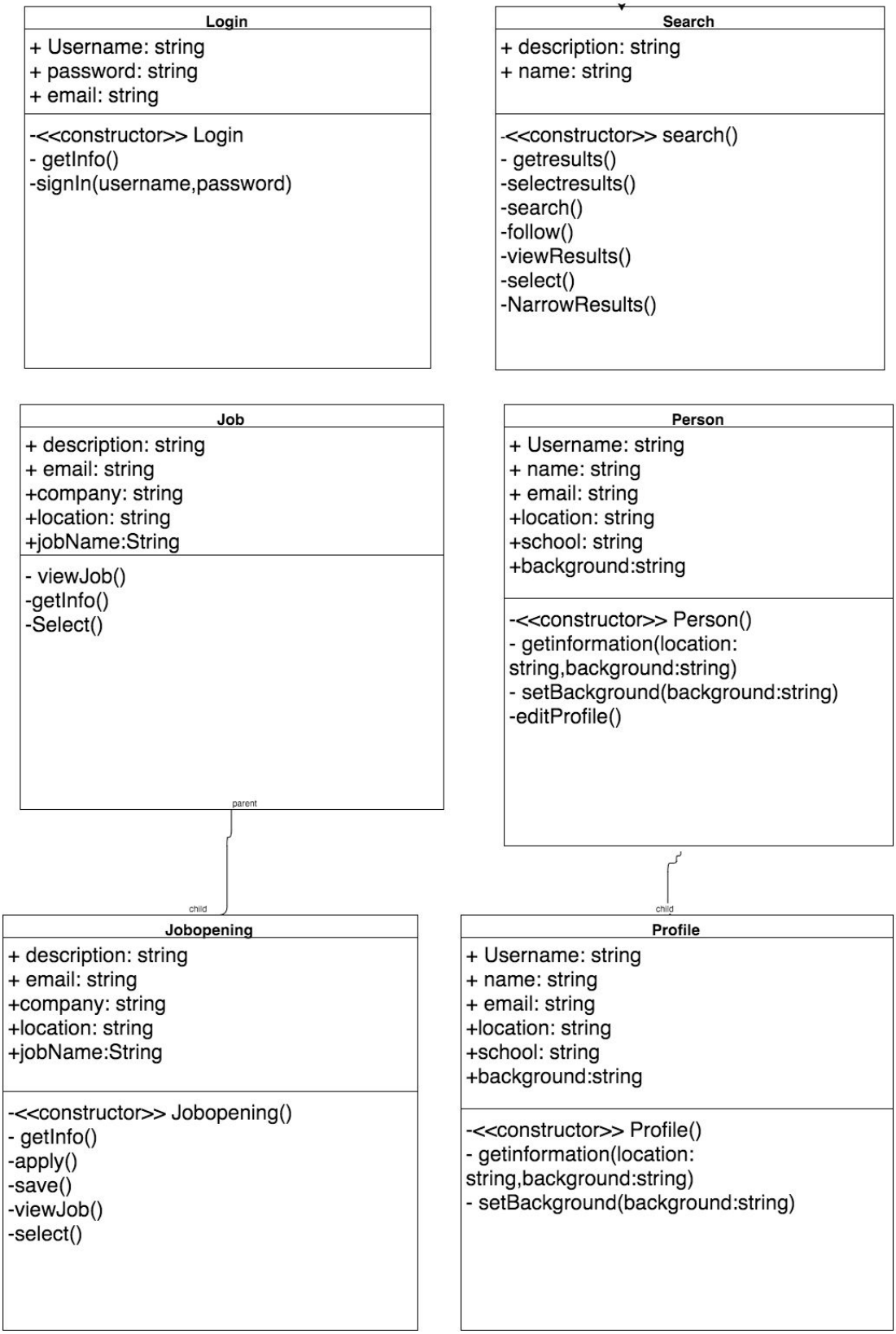
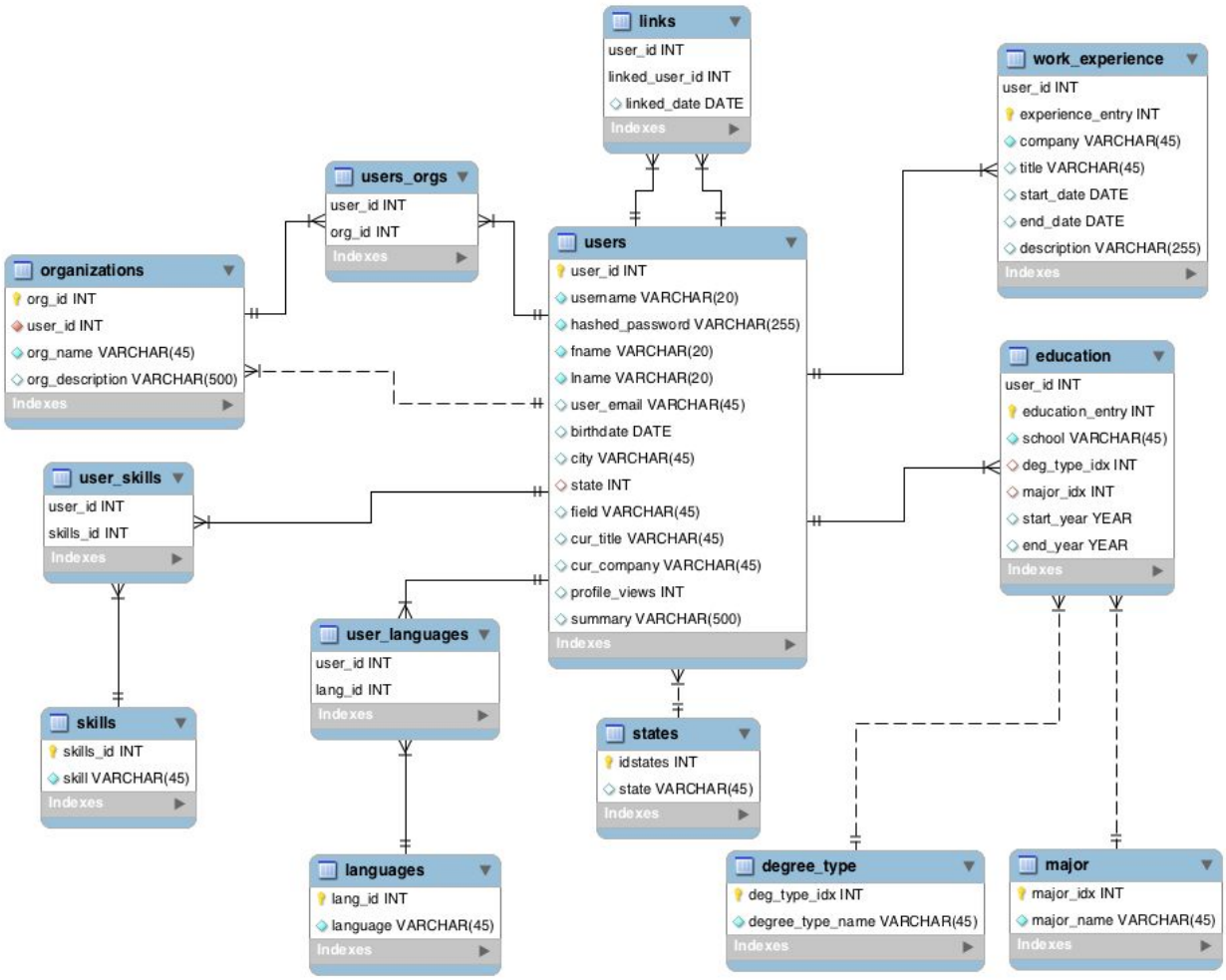


Table List (ERD)



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

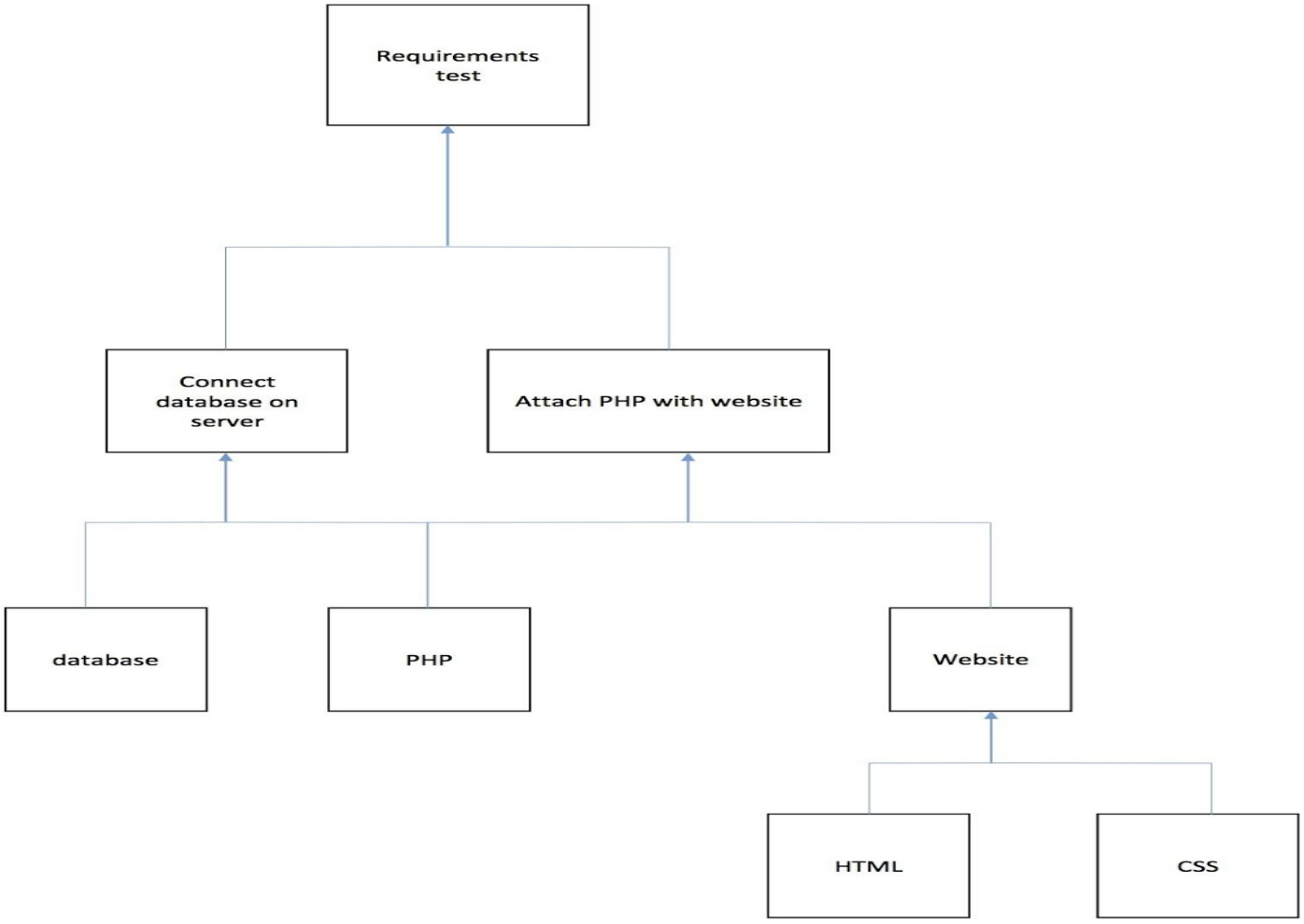
Testing Plan:

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
-- Fri Apr 22 21:38:07 2016
-- Model: New Model  Version: 1.0
-- MySQL Workbench Forward Engineering

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,
  `cur_title` VARCHAR(45) NULL,
  `cur_company` VARCHAR(45) NULL,
  `profile_views` INT NULL DEFAULT 0,
  `summary` VARCHAR(500) NULL,
  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;

-- -----------------------------------------------------
-- Data for table `linkedout`.`states`
-- -----------------------------------------------------
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (1, 'Alabama');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (2, 'Alaska');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (3, 'Arizona');
```

```
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (4, 'Arkansas');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (5, 'California');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (6, 'Colorado');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (7, 'Connecticut');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (8, 'Delaware');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (9, 'Florida');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (10, 'Georgia');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (11, 'Hawaii');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (12, 'Idaho');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (13, 'Illinois');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (14, 'Indiana');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (15, 'Iowa');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (16, 'Kansas');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (17, 'Kentucky');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (18, 'Louisiana');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (19, 'Maine');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (20, 'Maryland');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (21, 'Massachusetts');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (22, 'Michigan');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (23, 'Minnesota');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (24, 'Mississippi');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (25, 'Missouri');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (26, 'Montana');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (27, 'Nebraska');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (28, 'Nevada');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (29, 'New Hampshire');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (30, 'New Jersey');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (31, 'New Mexico');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (32, 'New York');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (33, 'North Carolina');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (34, 'North Dakota');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (35, 'Ohio');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (36, 'Oklahoma');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (37, 'Oregon');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (38, 'Pennsylvania');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (39, 'Rhode Island');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (40, 'South Carolina');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (41, 'South Dakota');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (42, 'Tennessee');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (43, 'Texas');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (44, 'Utah');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (45, 'Vermont');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (46, 'Virginia');
```

```
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (47, 'Washington');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (48, 'West Virginia');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (49, 'Wisconsin');
INSERT INTO `linkedout`.`states` (`idstates`, `state`) VALUES (50, 'Wyoming');

COMMIT;


-- -----------------------------------------------------
-- Data for table `linkedout`.`users`
-- -----------------------------------------------------
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`,
`lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`,
`profile_views`, `summary`) VALUES (1, 'user',
'$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'John',
'Smith', 'jsmith@gmail.com', '1900-01-01', 'Columbia', 25, 'Computer Science', 'Software
Engineer', 'University of Missouri', 8, 'about those TPS reports');
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`,
`lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`,
`profile_views`, `summary`) VALUES (11, 'user1',
'$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m',
'Natasha', 'Romanoff', 'blackwidow@avengers.com', '1984-11-22', 'New York', NULL,
'Super Hero', 'Black Widow', 'S.H.I.E.L.D./The Avengers', 5754, 'kicks ass and looks
good doing it');
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`,
`lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`,
`profile_views`, `summary`) VALUES (21, 'user2',
'$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Han',
'Solo', 'falcon@millenium.net', NULL, 'Outer Rim', NULL, 'Smuggling', 'Captain',
'self-employed', 1063, 'my son is super emo');
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`,
`lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`,
`profile_views`, `summary`) VALUES (31, 'user3',
'$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Boba',
'Fett', 'slave1@hutt.org', NULL, NULL, NULL, 'Bounty Hunting', 'Bounty Hunter',
'contracted', 2843, 'my backpack\'s got jets');
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`,
`lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`,
`profile_views`, `summary`) VALUES (41, 'user4',
'$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m',
```

'Obi-Wan', 'Kenobi', 'onlyhope@jedi.org', NULL, NULL, NULL, NULL, 'Force Ghost', '(deceased)', 54, NULL);
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`, `lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`, `profile_views`, `summary`) VALUES (51, 'user5', '$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Ren', '(unknown)', NULL, NULL, NULL, NULL, NULL, NULL, NULL, 4197, 'looking for my daddy.. i think i may have found him on an island');
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`, `lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`, `profile_views`, `summary`) VALUES (61, 'user6', '$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Ben', 'Solo', 'emokid@starkiller.gov', NULL, NULL, NULL, NULL, NULL, NULL, 48, NULL);
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`, `lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`, `profile_views`, `summary`) VALUES (71, 'user7', '$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Chuck', 'Bartowski', 'c.bartowski@buymore.com', NULL, NULL, 5, 'Spy/Nerd', 'Intersect', 'CIA/NSA', 207, NULL);
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`, `lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`, `profile_views`, `summary`) VALUES (81, 'user8', '$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Peter', 'Quill', 'starlord@gotg.org', NULL, NULL, NULL, NULL, NULL, NULL, 2491, NULL);
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`, `lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`, `profile_views`, `summary`) VALUES (91, 'user9', '$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Tony', 'Stark', 'iamironman@avengers.com', NULL, 'New York', NULL, 'Peace in our Time', 'Iron Man', 'Stark Industries/The Avengers', 976, 'excels at Hulk smashing, j/k.. we\'re buds!');
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`, `lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`, `profile_views`, `summary`) VALUES (101, 'user10', '$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Thor', 'Son of Odin', 'thunderstruck@avengers.com', NULL, 'Asgard', NULL, 'Weilding Mjolnir', 'God of Thunder', 'Paul Mitchell/The Avengers', 638, 'yes.. Mjolnir is a hammer. no.. i am not a carpenter');
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`, `lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`, `profile_views`, `summary`) VALUES (111, 'user11', '$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Steve', 'Rogers', 'cap@avengers.com', NULL, NULL, NULL, 'Shield Throwing Champion',

'Captain America', 'S.H.I.E.L.D./The Avengers', 25, 'seriously.. i can throw this thing so good...');
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`, `lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`, `profile_views`, `summary`) VALUES (121, 'user12', '$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Bruce', 'Banner', 'gammarays@avengers.com', NULL, NULL, NULL, 'Smashing', 'Incredible Hulk', 'The Avengers', 524, 'HULK SMASH!!!!');
INSERT INTO `linkedout`.`users` (`user_id`, `username`, `hashed_password`, `fname`, `lname`, `user_email`, `birthdate`, `city`, `state`, `field`, `cur_title`, `cur_company`, `profile_views`, `summary`) VALUES (131, 'user13', '$2y$10$oqkDqi7TY0ivbMAoJehnaOd3SfuzxMn1SL0sGKVkK3amSGXfJIG8m', 'Luke', 'Skywalker', 'lonelyjedi34@rebelalliance.org', NULL, 'Island Somewhere', NULL, 'Force Mastery', 'Last Jedi', 'unemployed', 681, 'i like long periods of isolation and staring into Rey\'s eyes for uncomfortably long periods of time');

COMMIT;

-- ---
-- Data for table `linkedout`.`work_experience`
-- ---
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`work_experience` (`user_id`, `experience_entry`, `company`, `title`, `start_date`, `end_date`, `description`) VALUES (1, 1, 'Univesity of Missouri', 'User Support Analyst', '2014-01-01', NULL, 'Tech support');

COMMIT;

-- ---
-- Data for table `linkedout`.`degree_type`
-- ---
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`degree_type` (`deg_type_idx`, `degree_type_name`) VALUES (1, 'Bachelor of Art');
INSERT INTO `linkedout`.`degree_type` (`deg_type_idx`, `degree_type_name`) VALUES (2, 'Bachelor of Science');
INSERT INTO `linkedout`.`degree_type` (`deg_type_idx`, `degree_type_name`) VALUES (3, 'Master');

```
INSERT INTO `linkedout`.`degree_type` (`deg_type_idx`, `degree_type_name`)
VALUES (4, 'PhD');
INSERT INTO `linkedout`.`degree_type` (`deg_type_idx`, `degree_type_name`)
VALUES (5, 'MD');
INSERT INTO `linkedout`.`degree_type` (`deg_type_idx`, `degree_type_name`)
VALUES (6, 'JD');
INSERT INTO `linkedout`.`degree_type` (`deg_type_idx`, `degree_type_name`)
VALUES (7, 'MBA');

COMMIT;


-- -----------------------------------------------------
-- Data for table `linkedout`.`major`
-- -----------------------------------------------------
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`major` (`major_idx`, `major_name`) VALUES (1, 'Computer
Science');
INSERT INTO `linkedout`.`major` (`major_idx`, `major_name`) VALUES (2, 'Information
Technology');
INSERT INTO `linkedout`.`major` (`major_idx`, `major_name`) VALUES (3, 'Business');

COMMIT;


-- -----------------------------------------------------
-- Data for table `linkedout`.`education`
-- -----------------------------------------------------
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`education` (`user_id`, `education_entry`, `school`,
`deg_type_idx`, `major_idx`, `start_year`, `end_year`) VALUES (1, 1, 'University of
Missouri', 2, 1, 2013, 2017);
INSERT INTO `linkedout`.`education` (`user_id`, `education_entry`, `school`,
`deg_type_idx`, `major_idx`, `start_year`, `end_year`) VALUES (1, 2, 'Harvard
University', 3, 1, 2017, 2020);

COMMIT;


-- -----------------------------------------------------
-- Data for table `linkedout`.`skills`
```

```
-- -----------------------------------------------------
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`skills` (`skills_id`, `skill`) VALUES (1, 'MS Office');
INSERT INTO `linkedout`.`skills` (`skills_id`, `skill`) VALUES (2, 'OS X');

COMMIT;


-- -----------------------------------------------------
-- Data for table `linkedout`.`user_skills`
-- -----------------------------------------------------
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`user_skills` (`user_id`, `skills_id`) VALUES (1, 1);
INSERT INTO `linkedout`.`user_skills` (`user_id`, `skills_id`) VALUES (1, 2);

COMMIT;


-- -----------------------------------------------------
-- Data for table `linkedout`.`languages`
-- -----------------------------------------------------
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`languages` (`lang_id`, `language`) VALUES (1, 'English');
INSERT INTO `linkedout`.`languages` (`lang_id`, `language`) VALUES (2, 'Spanish');
INSERT INTO `linkedout`.`languages` (`lang_id`, `language`) VALUES (3, 'French');
INSERT INTO `linkedout`.`languages` (`lang_id`, `language`) VALUES (4, 'German');

COMMIT;


-- -----------------------------------------------------
-- Data for table `linkedout`.`user_languages`
-- -----------------------------------------------------
START TRANSACTION;
USE `linkedout`;
INSERT INTO `linkedout`.`user_languages` (`user_id`, `lang_id`) VALUES (1, 1);
INSERT INTO `linkedout`.`user_languages` (`user_id`, `lang_id`) VALUES (1, 2);

COMMIT;
```

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` = ?);
```

Sprint 3: Development

Luke:

Polished the code of the webpages and oversaw development.
Git configuration and development practices tutorial.

Everyone will fork the master branch of the project and commit changes in their own fork. Once they've completed testing on their own Azure web server, they will submit a pull request to merge their updates back into the master branch.

Mark:

HTML templates, with base PHP, for user login and registration (index.php, register.php)
HTML template for ViewProfile.php

Scott:

Reviewing and confirming pull requests to master branch.
PHP scripting for login, registration and logout (index.php, register.php, logout.php)

Integration with database and testing.
Security using PHP $_SESSION variables.
Header redirects based on state.

Review ViewProfile.php and editprofile.php

Edits to PHP scripts and queries.
Additional functionality in navbar and PHP

Eric:

PHP scripting for ViewProfile.php

SQL queries to build page based on profile requested.

Review HTML template

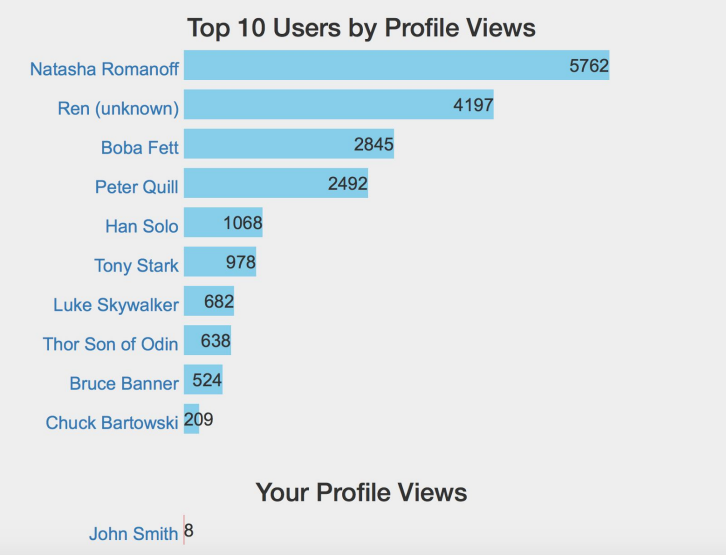
Minor changes for the HTML template

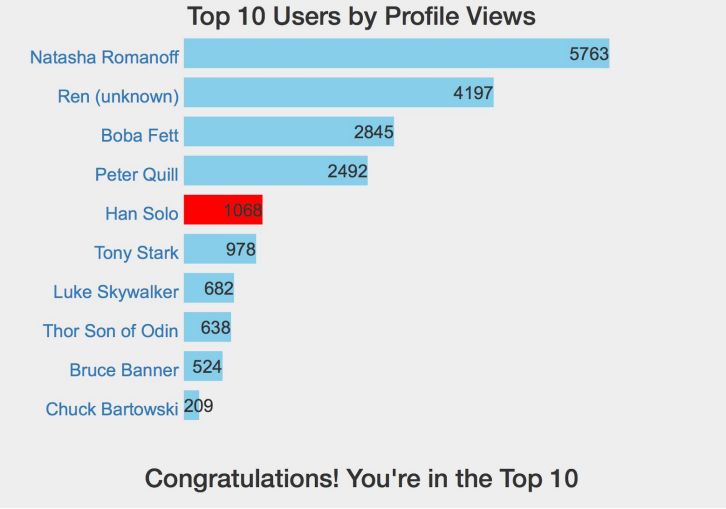
Yaoxuan:

PHP scripting for editprofile.php (work in progress)

Data Visualization

We are planning on using the number of profile views users receive to create a “Top 10” chart. If a user is not in the “Top 10” they will also see their personal rank. Rather than running a script that blindly increments profile_view whenever a profile page is viewed we will want to check to make sure that the logged in user doesn’t match the profile being viewed. This way a person cannot game the system by repeatedly viewing their own profile. The “Top 10” will be a separate page in the app which queries the database to build the list. Users will also be able to click on “Top 10” users to go view their profile.





Edge Cases

Case 1: There are no registered users to the app.

In its current form the app will be completely secured behind login. Therefore pages cannot be accessed without a registered user. If there are no registered users at some point in time, the app cannot be accessed.

For Top10 data visualization, if there are less than 10 registered users, the chart will show only the number of users there are. It just looks odd to say “Top 10” at the top of the screen if there are less than 10 users in the chart.

Case 2: A user has no linked contacts.

After logging in, a user can look at their list of linked contacts. If they have none, they will simply get a friendly message that they don’t have any connections yet and some instruction to help them get started finding people to connect with.

Case 3: Too many users accessing app simultaneously.

The database currently has a limit of 4 simultaneous connections. This is the free version of the MySQL DB through Microsoft Azure because it wouldn’t let us upgrade to the next tier (with a limit of 10 connections) without adding a credit card. If more users are trying to use the app than the database can accept connections from, all users over the limit will not get connected to the database and will just see empty pages. Besides the connection limit, the size of the database is extremely limited (0.02 GB) in the free version.

Sprint 4: Development

Luke:

Working on searching for users and connecting with them

Mark:

Bootstrap HTML template for connections.php
Overall CSS and Bootstrap styling

Scott:

Top10.php data visualization
connections.php scripts
Working on editprofile.php
Bootstrap Navbar unification

Eric:

Working on editprofile.php

Yaoxuan:

Working on editprofile.php

Sprint 4: Issues

1. When clicking to ‘View Profile’ from connections.php, ViewProfile.php sometimes isn’t loading correctly.
    a. SOLVED - user.state in database cannot be NULL, the ViewProfile select statement breaks. Made user registration insert ‘blank’ state id into new user row
2. editprofile.php has proven to be really complicated. It’s taken up a lot of development time.
    a. SOLVED - inserts and updates are functional. Further testing of form inputs and adding more profile information would be needed if time allowed.

Next Steps

Current Needs:

- Make Connections a request/confirm handshake agreement rather than automatic.
- Further testing of profile editing to ensure functionality.
- Set up and enforce HTTPS.

Future Development:

- Add more profile sections to website. Database currently includes tables for user languages and skills (both not implemented).
- Add organization functionality.
- Add communication for user/organizations via messages and/or posts