SW Engineering CSC648/848 Spring 2021

SkillSeek

Section 02 Team 01 Members:
Emin Musayev
Ramzi Abou Chahine
Begum Sakin
Ali Bin Sabir
Melissa Gonzalez

3/9/2021

Table of Contents

Functional Requirements

User Independent Functionality

1. Users shall be able to register for accounts. (Priority 1)
2. Users shall receive a verification email after account creation. (Priority 1)
3. Users shall be able to log in and out safely. (Priority 1)
4. Users shall be able to edit their profile and password. (Priority 1)
5. Users shall be able to search for other users and job listings on the platform. (Priority 1)
6. Users shall be able to view other users' profiles. (Priority 1)
7. Users shall be able to connect with other users by sending them a request. (Priority 2)
8. Users shall be able to use boolean search to refine their search results. (Priority 3)

Student Functionality

1. Students shall be able to include their name, a short biography, their location, demographic, gender, current profession, and date of birth on their profiles. (Priority 1)
2. Students shall be able to upload a profile picture and resume on their profiles. (Priority 1)
3. Students shall be able to list education and work experiences on their profile. (Priority 2)
4. Students shall be able to receive reviews from teachers. (Priority 1)
5. Students shall be able to apply to view and apply for job listings. (Priority 2)

Teacher Functionality

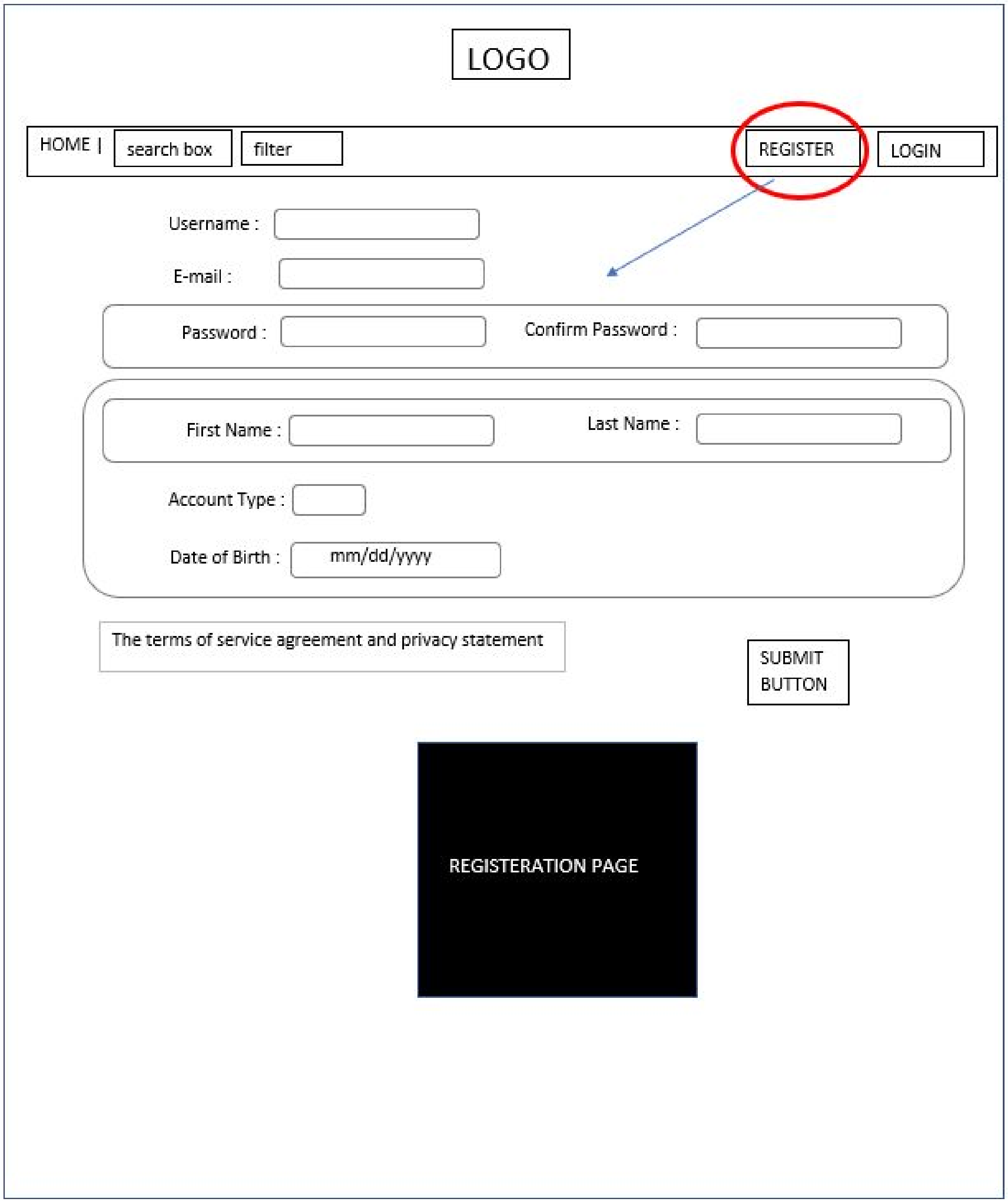
1. Teachers shall be able to include their name, a short biography, their location, demographic, gender, current profession, and date of birth on their profiles. (Priority 1)
2. Teachers shall be able to upload a profile picture and resume on their profiles. (Priority 1)
3. Teachers shall be able to list education and work experiences on their profile. (Priority 2)
4. Teachers shall be able to review students for potential employers to view. (Priority 1)
5. Teachers shall be able to view job listings (Priority 2)

Employer Functionality

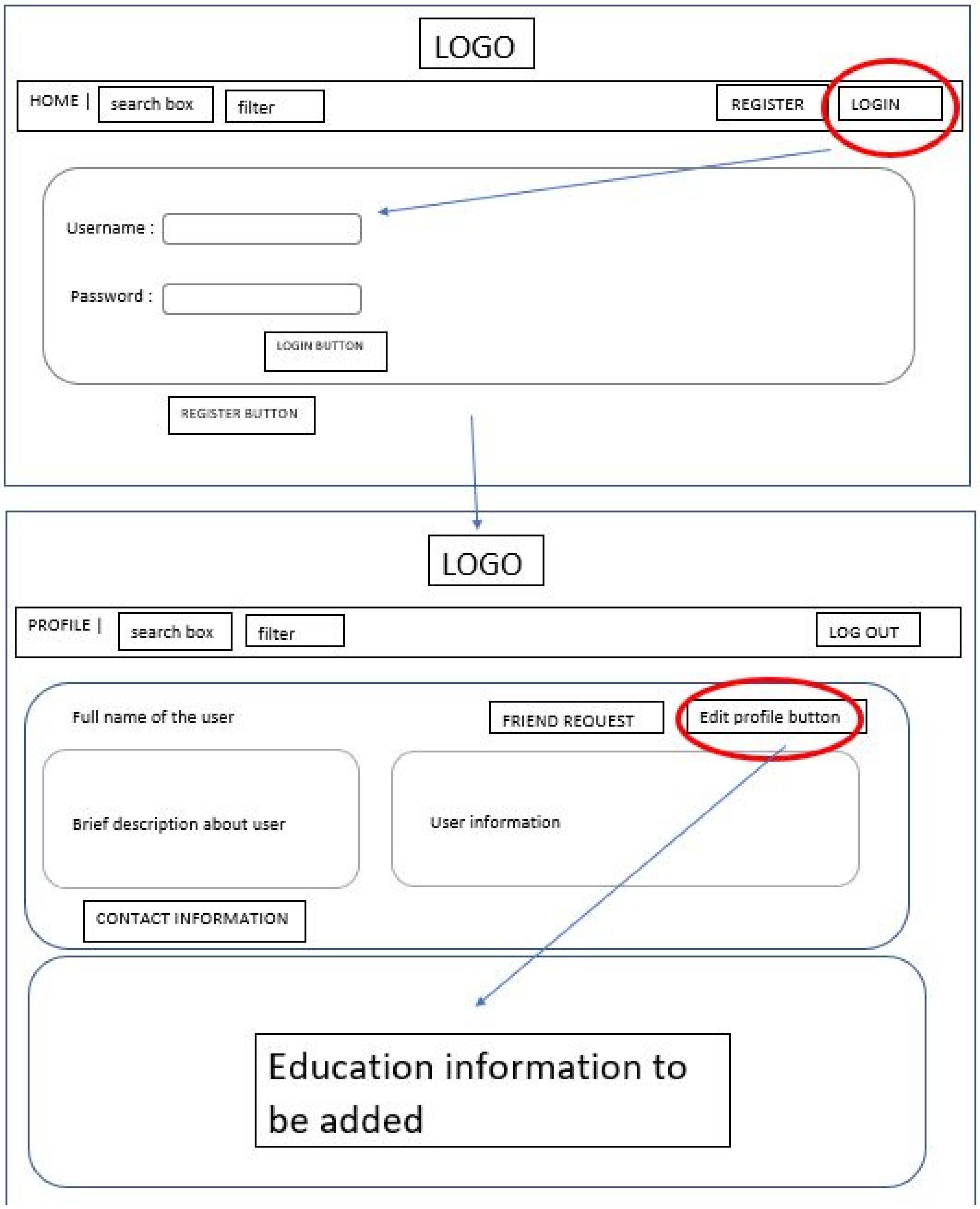
1. Employers shall be able to include their organization's name, a short description of their corporation, a logo, and the location of their headquarters. (Priority 1)
2. Employers shall be able to post multiple job listings for potential candidates to apply for. (Priority 1)
3. Employers shall be able to include a job title, description, job location, and an expiry date for the job listing. (Priority 1)
4. Employers shall be able to contact applicants via email. (Priority 1)
5. Employers shall be able to delete and hide posted job listings. (Priority 3)
6. Employers shall be able to view candidates who applied for a job listing while the listing is not expired and for a period of 2 months after it has expired. (Priority 1)

UI Mockups and Storyboards

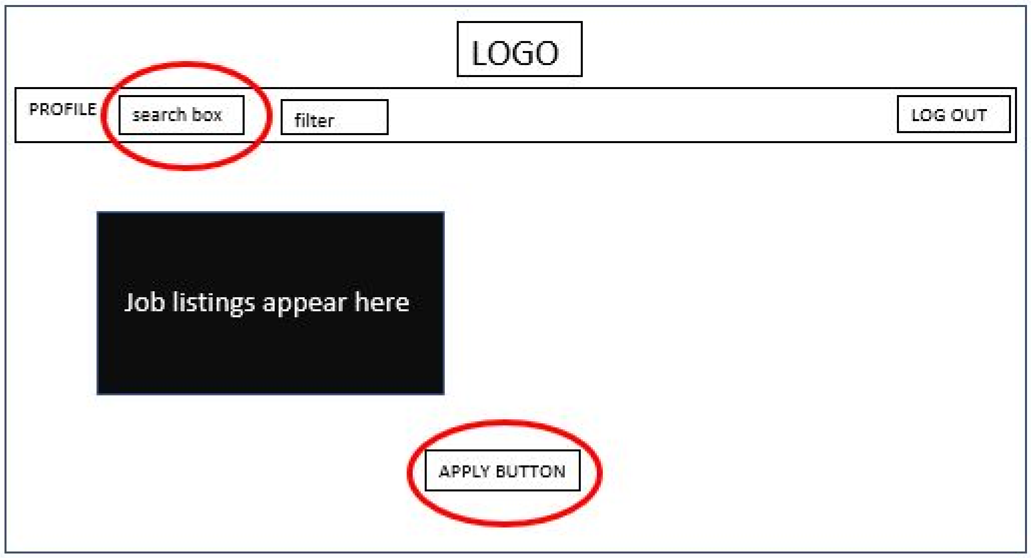
Use Case 1 - Student Finding A Job:



The student registers an account.

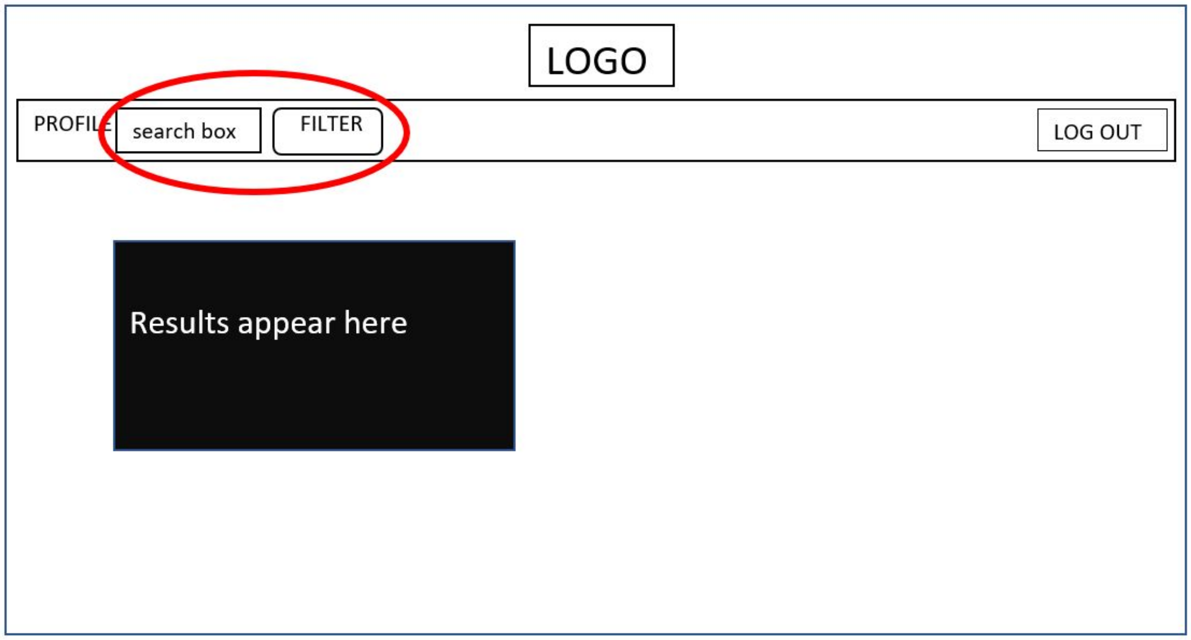


The student logs in to their account and adds/edits education information on their profile.

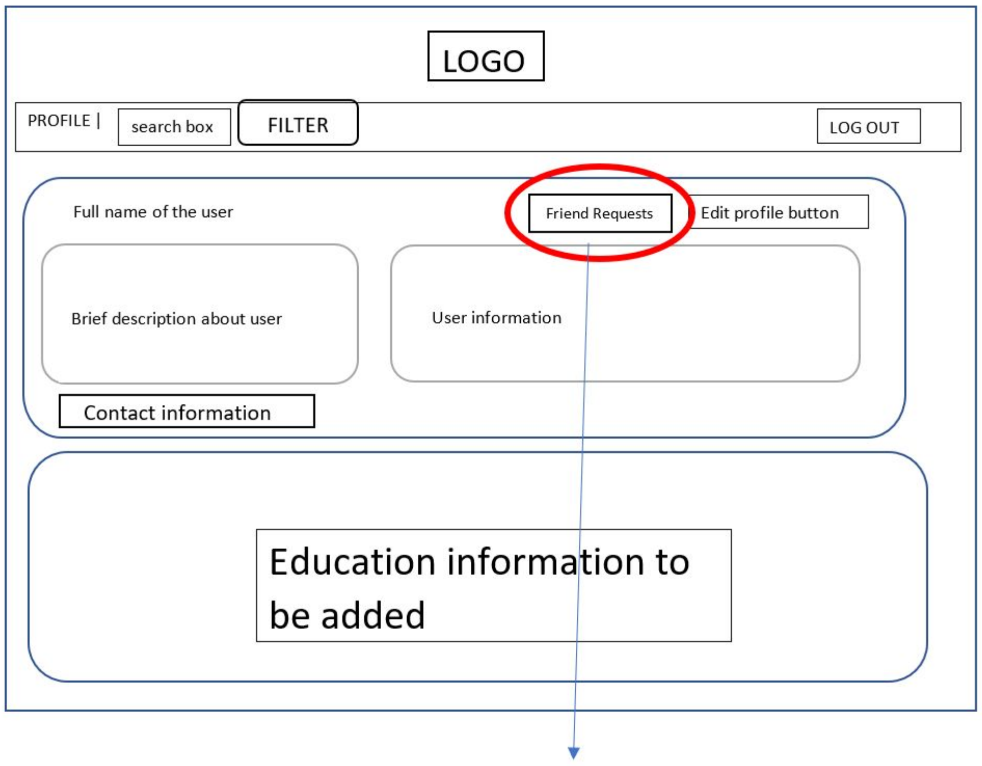


The student utilizes the search bar to look for jobs that they are interested in and then the student hits the apply button. The student must utilize the ‘FILTER’ feature and set it to ‘Job’ for better results.

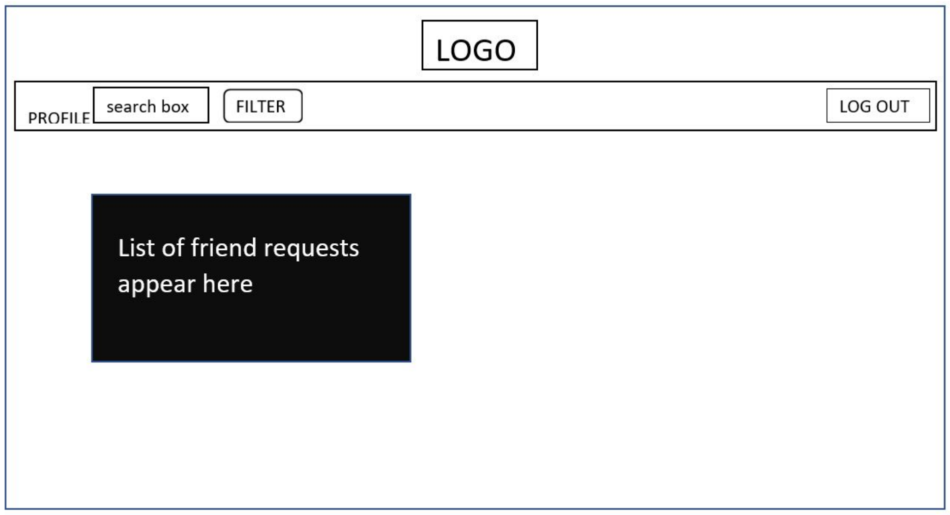
Use Case 2 - Student Finding A Friend:



The student utilizes the search bar to find friends they wish to connect with. Friends can be found by inputting names or their unique username. The student must utilize the 'FILTER' feature and set it to 'Student' for better results. A student can view their friend's contact information on their page only after they have connected mutually.

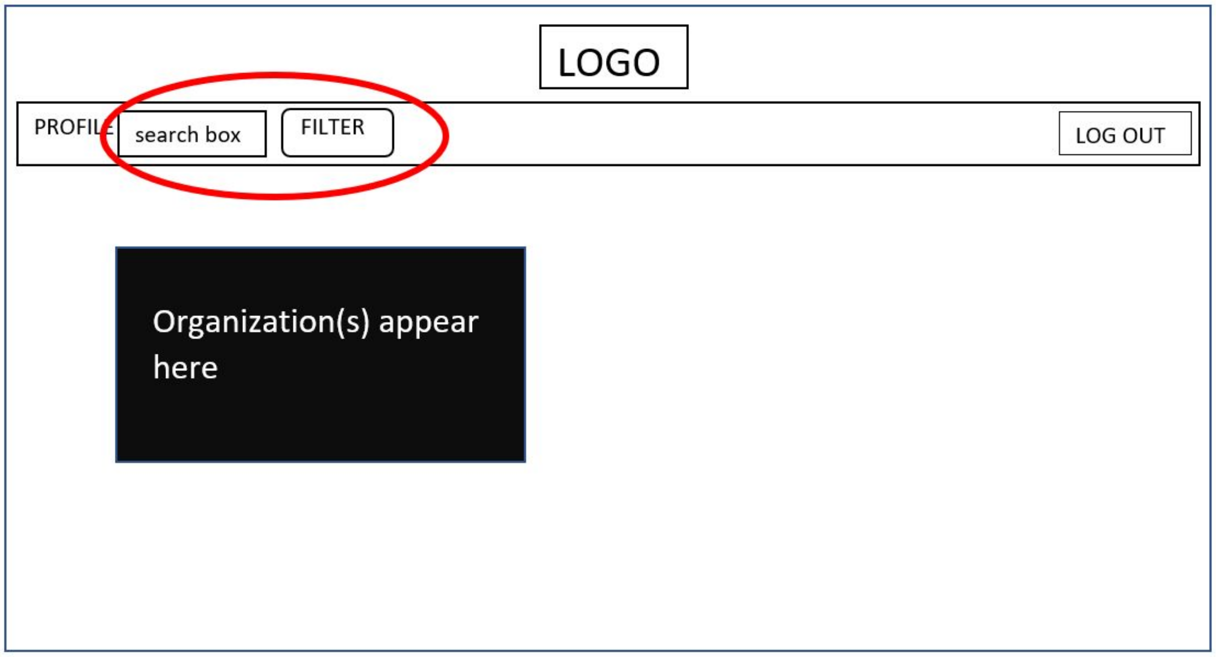


The student goes to their profile page where they are able to check any pending friend request they may have.



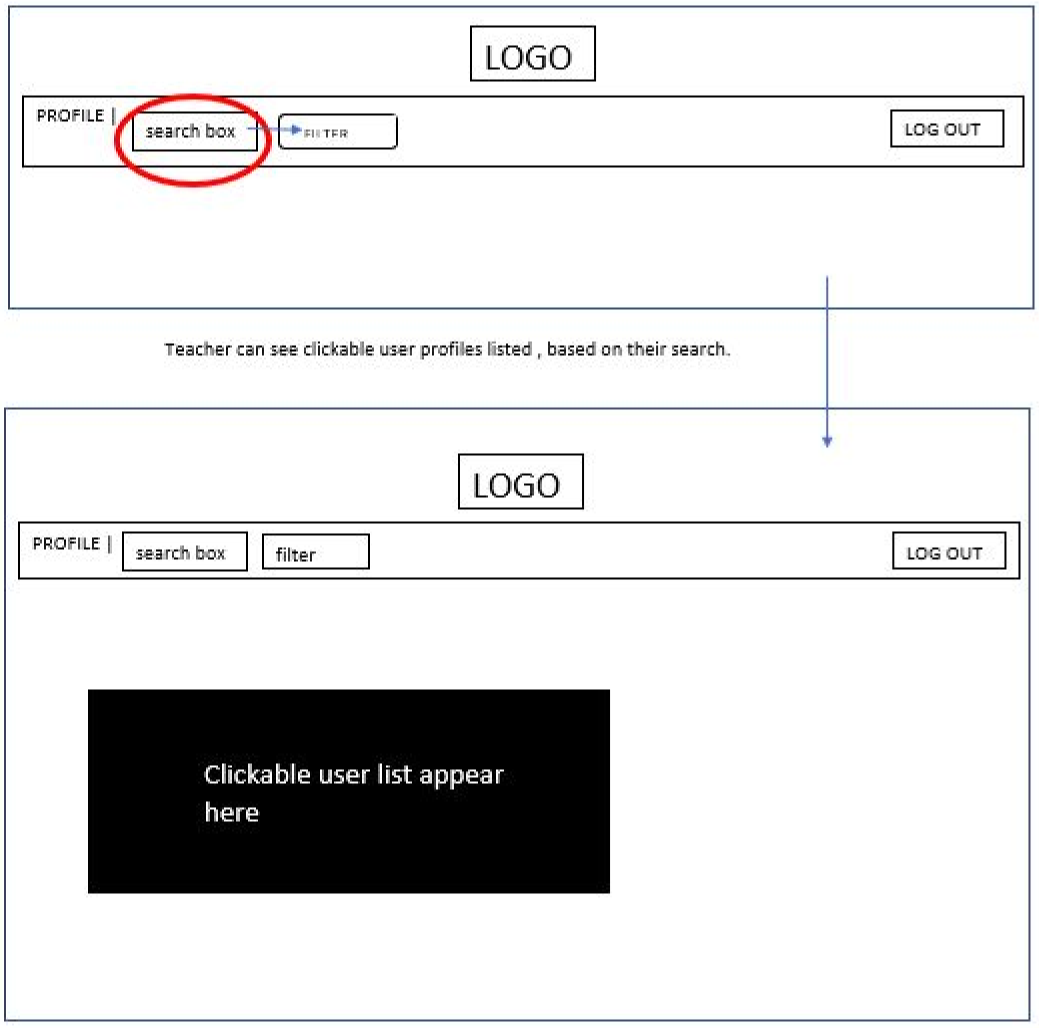
Once the ‘Friend Requests’ button is clicked, it will take them to a page where all (if any) of their pending friend requests will be listed. They will be able to accept or decline each request on this page.

Use Case 3 - Student Finding An Organization:

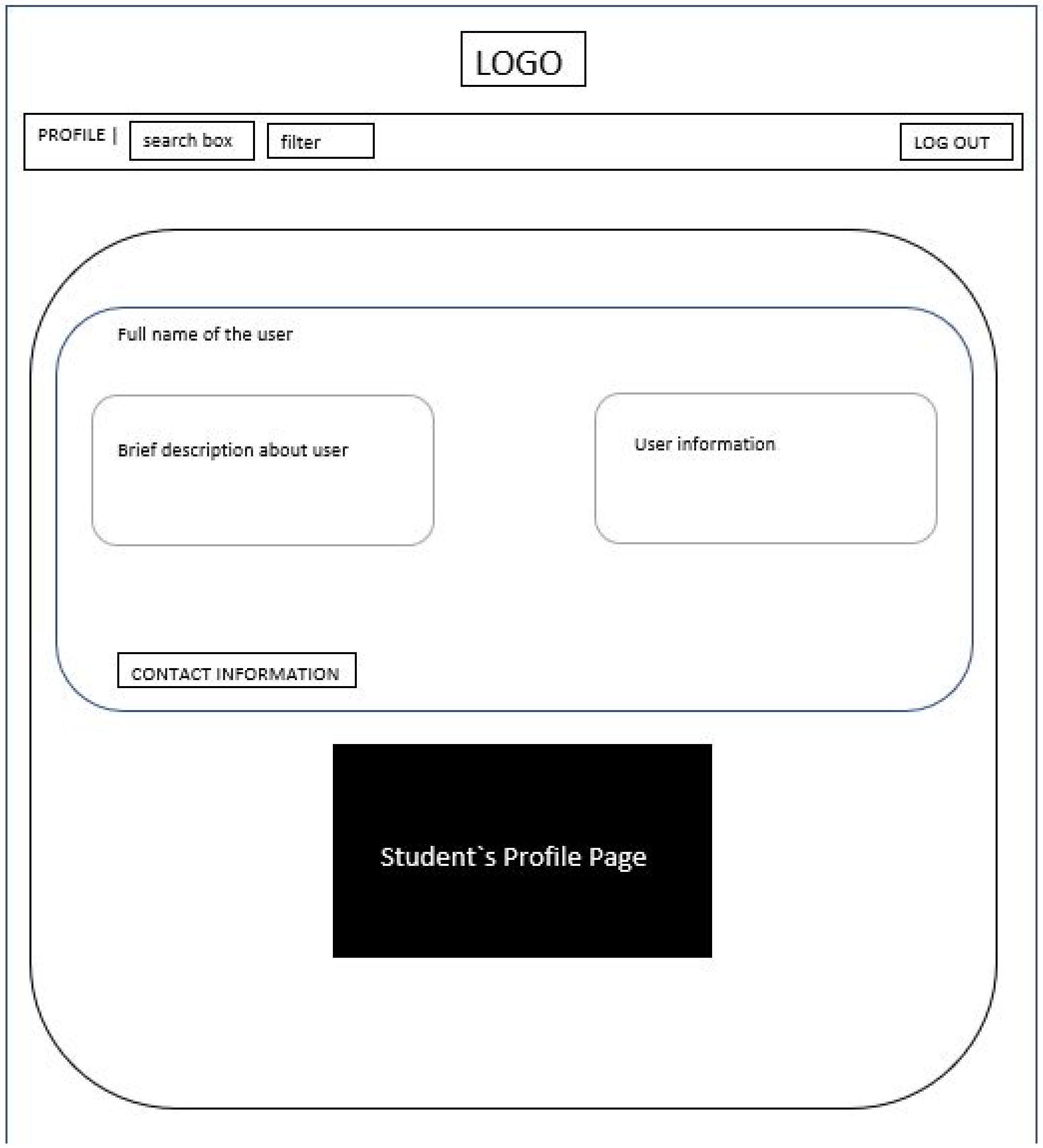


The student utilizes the search bar to find organizations they wish to look up or follow. The student must utilize the 'FILTER' feature and set it to 'Organization' for better results.

Use Case 4 - Teacher Finding/Reviewing Student:

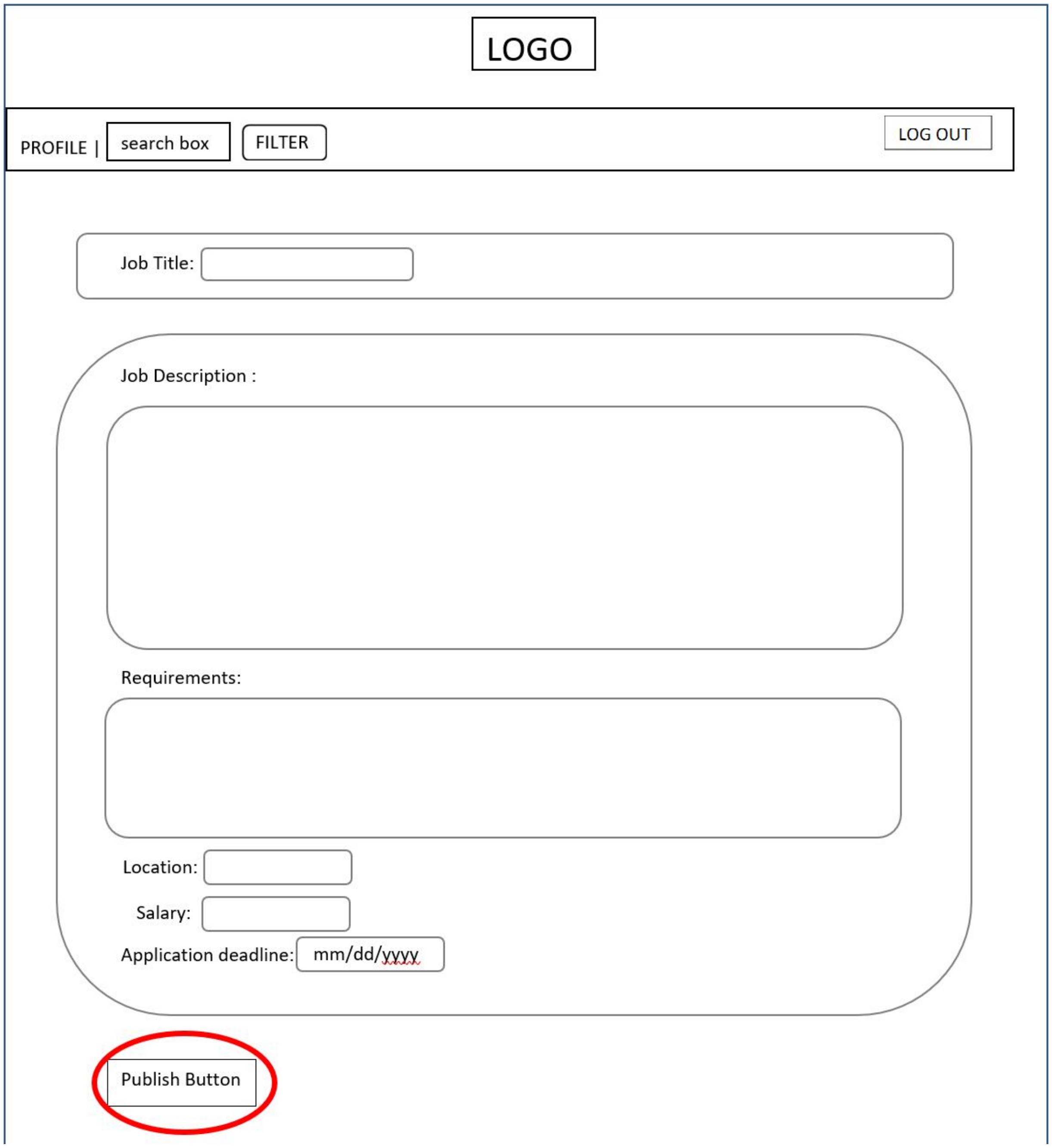

LOGO
PROFILE |
search box
FILTER
LOG OUT
Teacher can see clickable user profiles listed , based on their search.
LOGO
PROFILE |
search box
filter
LOG OUT
Clickable user list appear here

Teacher logs into their account and sets the search box filter to “user” for finding a student. Teachers can see clickable user-profiles listed, based on their search.

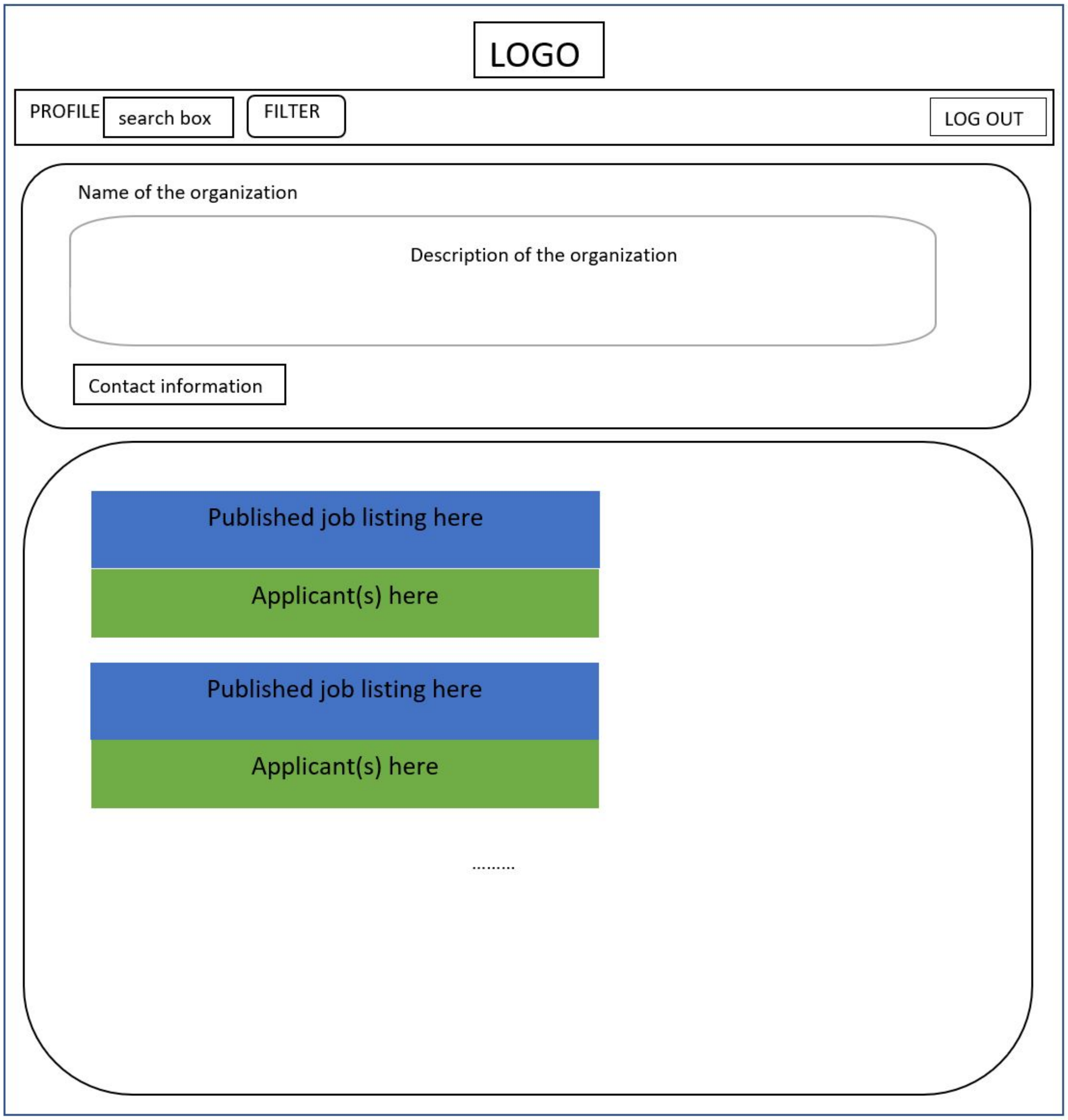


The teachers can click on the profiles and review the information. (They will see contact information IF connected with the student.)

Use Case 5 - Employer Looking For Employees:



An employer will be able to create a job listing that they wish to publicize and offer to any user looking for that exact job or who simply wish to work for them.



The employer will be able to go to their page and see every job listing that they have made public. Below every listing are the applicant(s) who have made a submission and are interested in the position. The employer will be able to look at applicants and applications conveniently.

High-Level Architecture and Database Organization

Database Organization

Our database must be designed to track several entities and their relationship with one another. The most basic entity will be users and their associated information. Users will differ based on the account type, of which there will be three, students, teachers, and employers.

User Entity Attributes - users:

- User ID (Primary Key) - uid - BIGINT
- Username (Unique) - username - VARCHAR(64)
- Password - password - VARCHAR(64)
- First Name - firstName - VARCHAR(128) (will be treated as organization name for employers)
- Last Name - lastName - VARCHAR(128) (NULL for employers)
- Email (Unique) - email - VARCHAR(255)
- Date Of Birth - dateOfBirth - DATE (will be treated as date of establishment for employers)
- Account Type - accType - VARCHAR(10)
- Verified - verified - TINYINT

Each user must have a profile page to host personal information about themselves or their organization. Here we must distinguish between account types in order to host the appropriate information for user profiles. As attributes will vary, we shall create two separate entities to host profile data; one entity to store information belonging to students and teachers and another to store that of the employer. Each user shall have exactly one profile.

A blueprint for the entities and their attributes shall look like the following:

Individual Profiles - profileIndividual: (Student & Teacher)

- User ID (Primary Key & Foreign Key) - fk_uid - BIGINT
- Profile Picture - profile picture - VARCHAR(4096)

- Resume - resume - VARCHAR(4096)
- Current Profession - currentProfession - VARCHAR(128)
- Description - description - VARCHAR(4096)
- Gender - gender - VARCHAR(12)
- Location - location - VARCHAR(128)
- Demographic - demographic - VARCHAR(48)

Organizational Profiles - profileOrg: (Employer)

- User ID (Primary Key & Foreign Key) - fk_uid - BIGINT
- Logo - logo - VARCHAR(4096)
- Headquarters Location - location - VARCHAR(128)
- Description - description - VARCHAR(4096)

To provide users with the ability to display their education and work-related experiences we need two separate entities to house the associated data. Both students and teachers may find uses for these features, so they shall have the ability to include this information on their profile whereas, employers shall not. The distinction made here is simply a design choice as the employer account type represents organizations such as businesses, corporations, institutions, etc. Each eligible user shall have as many educational and work-related experiences associated with their account as they desire. For future reference, we will distinguish between these two entities using the following terms: education for educational experience and experience for the work-related experience.

Education Entity - education:

- Education ID (Primary Key) - education_id - BIGINT
- User ID (Foriegn Key) - fk_uid - BIGINT
- Degree Type - degree_type - VARCHAR(32)
- Degree Name - degree_name - VARCHAR(128)
- University - university - VARCHAR(128)
- Start Date - beginDate - DATE
- End Date (can be NULL) - endDate - DATE
- Ongoing - ongoing - TINYINT (to distinguish between past and present educational experiences)

Experience Entity - experience:

- Experience ID (Primary Key) - experience_id - BIGINT
- User ID (Foriegn Key) - fk_uid - BIGINT
- Employer - employer - BIGINT
- Job Title - experience - VARCHAR(128)
- Description - description - VARCHAR(512)
- Start Date - beginDate - DATE
- End Date (can be NULL) - endDate - DATE
- Ongoing - ongoing - TINYINT (to distinguish between past and present experiences)

To provide teachers with the capability of adding reviews to student profiles, we need to account for another entity which shall be referred to as ratings. Reviews shall only appear on student profiles as they are the only account type eligible for reviews. Teachers shall be able to post any number of reviews for any individual student account and shall be able to review as many students as they desire.

Review Entity - rating:

- Review ID (Primary Key) - rating_id - BIGINT
- Review Title - title - VARCHAR(64)
- Review - review - VARCHAR(4096)
- Rating - rating - TINYINT
- Reviewee User ID (Foreign Key) - rater - BIGINT
- Reviewer User ID (Foreign Key) - rated - BIGINT
- Date Reviewed - dateRated - DATE

As for organizational profiles, they are distinct since they shall act as a portal of sorts to the employer's available job listings. To allow employers to create job listings for students and teachers to apply to, a new entity is required. The job listing entity will have several vital attributes associated with it to allow for this functionality.

Job Listing Entity - jobListing:

- Listing ID (Primary Key) - listing_id - BIGINT
- Poster User ID (Foriegn Key) - poster_id - BIGINT
- Job Title - jobTitle - VARCHAR(64)
- Description - description - VARCHAR(4096)
- Location - location - VARCHAR(128)
- Expiry Date - expiryDate - DATE
- Expired - expired - TINYINT
- Hidden - hidden - TINYINT

A user-defined expiry date attribute shall allow the job listing to expire and be hidden from all other users. When the expiry date is reached, expiry shall be set to true. The hidden attribute shall allow employers to voluntarily hide a job listing from view without permanently deleting it. Expired job listings shall be preserved for a period of 60 days, so employers have the opportunity to observe the profiles of applicants and select candidates. After the job listing is deleted, all child entities shall be deleted such as job applicants.

Job Applicant Entity - jobApplicant:

- Application ID (Primary Key) - application_id - BIGINT
- Listing ID (Foreign Key) - listing_id - BIGINT
- Applicant User ID (Foreign Key)- applicant_uid - BIGINT

Another feature we shall implement is the ability to follow organizations and befriend other users. This feature requires more discussion to decide on how users shall be able to interact with one another. Regardless of the design choices made, the underlying entity that allows for such an interaction should remain largely unchanged. The connections entity is responsible for this interaction.

Connections Entity - connections:

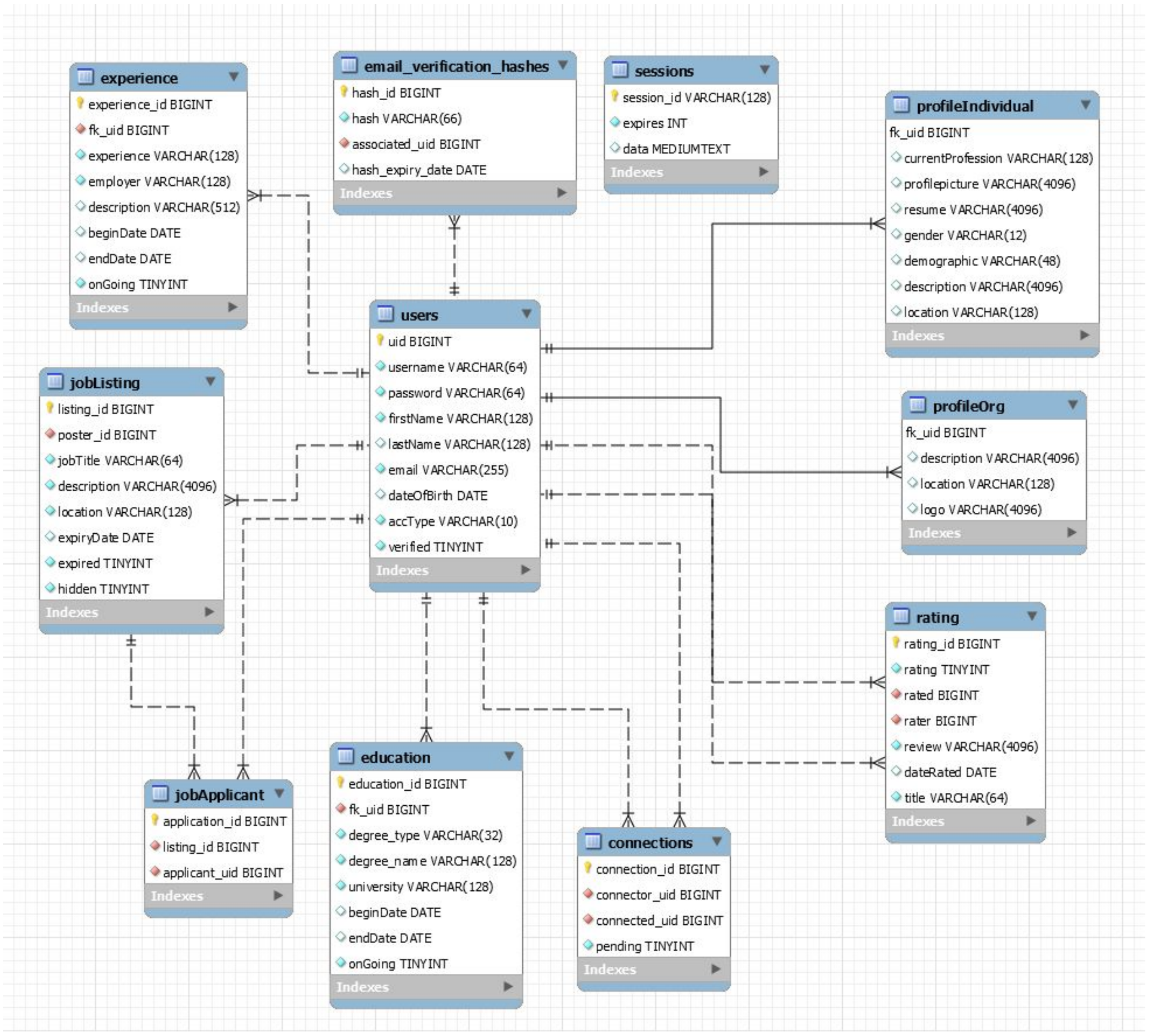
- Connections ID (Primary Key) - connections_id - BIGINT
- Connector ID (Foriegn Key) - connector_uid - BIGINT
- Connected ID (Foriegn Key) - connected_uid - BIGINT
- Pending - pending - TINYINT

To enforce email verification, we require an entity associated with each created account following the completion of the registration process. After the verification is completed the entity associated with the user shall be deleted and the verification status of the said user shall be updated. Accounts that are not verified within 2 days shall be deleted.

Email Verification Entity - email_verification_hashes:

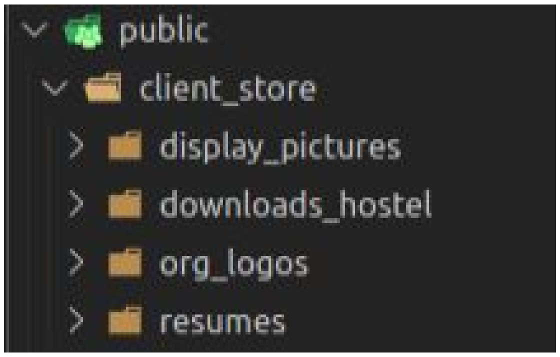
- Hash ID (Primary Key) - hash_id - BIGINT
- Hash - hash - VARCHAR(66)
- Associated User ID (Foreign Key) - associated_uid - BIGINT
- Expiry Date - expiryDate - DATE

Finally, an entity named sessions shall be automatically generated by, express-sessions, a module we are using to handle user sessions.



Media Storage

User-uploaded media such as resumes, profile pictures, and logos shall be stored in the filesystem. A relative path to the files associated with a user shall be recorded in the database.



All user-uploaded data will be stored in the client_store directory. When a file is uploaded by a user, it is downloaded to the downloads_hostel directory. Depending on the type of file uploaded, the file shall then be moved to an appropriate directory and the path to the file shall be stored in the database as such:

/public/client_store/resumes/*randombytes*.pdf

Search/Filter Architecture and Implementation

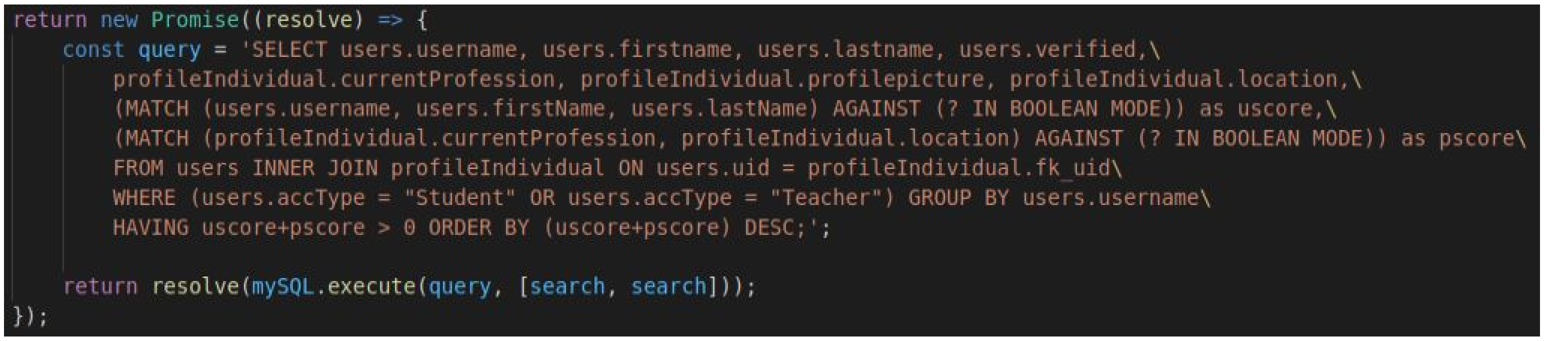
To search the database for information relevant to the user, a search box with three filters shall be implemented, Users, Employers, and Job Listings. Searching with the user filter selected shall return results on registered students and teachers. Searching with the employer filter shall return results on registered employers. Finally, searching with the job listings filter shall return results on active and unhidden job listings.

Searching for users can be carried out by seeking a user's username, first name, last name, full name, current profession, and location. The tables that host this information are users and profileIndividual.

Searching for employers can be carried out by seeking an employer's username, organization name, and location. The tables that host this information are users and profileOrg.

Searching for job listings can be carried out by seeking a job listing's poster's username, posters's organization name, job title, and job location. The tables that host this information are users and jobListing.

Searching shall be implemented using a MySQL statement using the MATCH() and AGAINST() keywords to retrieve sorted results directly from the database. The database shall sort results according to relevance to the searched query by using the value returned by MATCH and AGAINST to determine the order. An example of this concept is demonstrated below with the user search query: (‘?’ is the search term)



```
return new Promise((resolve) => {
    const query = 'SELECT users.username, users.firstname, users.lastname, users.verified,\
        profileIndividual.currentProfession, profileIndividual.profilepicture, profileIndividual.location,\
        (MATCH (users.username, users.firstName, users.lastName) AGAINST (? IN BOOLEAN MODE)) as uscore,\
        (MATCH (profileIndividual.currentProfession, profileIndividual.location) AGAINST (? IN BOOLEAN MODE)) as pscore\
        FROM users INNER JOIN profileIndividual ON users.uid = profileIndividual.fk_uid\
        WHERE (users.accType = "Student" OR users.accType = "Teacher") GROUP BY users.username\
        HAVING uscore+pscore > 0 ORDER BY (uscore+pscore) DESC;';

    return resolve(mySQL.execute(query, [search, search]));
});
```

Documentation of API

All visitors

1. / - GET
   - Displays home page with most recent job listings added to the platform.
   - Response Types: application/json
     a. Code : 200
     ```
     [
       {
         “listing_id” : 1,
         “jobTitle” : “Junior Front End Developer”,
          “location” : “San Francisco, CA, USA”,
          “expiryDate” : “03-07-2021”,
          “expired” :   0,
          “hidden” : 0,
          “logo” : “/public/client_store/logos/2133420847.png”,
          “username” : “skillseek”,
          “firstName” : “SkillSeek”
       },
       ...
     ]
     ```
     b. Code: 301 (Redirect to enforce HTTPS)

2. /login - GET
   - Displays login page with form that must be submitted to login.

3. /registration - GET
   - Displays registration page with a sign up form.

4. /users/login - POST
    - Submit login form and complete send request.
    - Parameters:
        a. Username
        b. Password
    - Response Types:
        a. Code: 302 (Redirect on successful login)
        b. Code: 200 (Invalid credentials)
        c. Code: 500 (Server Error)

5. /users/registration - POST
    - Submit registration form and complete send request.
    - Parameters:
        a. Username
        b. Email
        c. First Name
        d. Last Name
        e. Date Of Birth
        f. Permission For Cookies Usage
        g. Acceptance of Terms Of Service and Privacy Policy
    - Responses Type:
        a. Code: 302 (Redirect on successful login)
        b. Code: 200 (Invalid credentials)
        c. Code: 500 (Server Error)

6. /verify - GET
    - Form submitted via email.
    - Parameters:
        a. Hash
    - Response Types:
        a. Code: 302 (Redirect on successful verification)
        b. Code: 200 (Account could not be verified)

All users

1. /profile OR /profile/{username} - GET
   - View user profile.
   - Response Types: application/json (depends on user profile type)
     a. Code: 200 (student profile)

```
[
  “uid” : 100,
  “username” : “ramzi”,
  “firstName” : “Ramzi”,
  “lastName” : “Abou Chahine”,
  “dateOfBirth”: “11-30-2020”,
  “accType” : “Student”,
 “currentProfession” : “Computer Science Student”,
 “profilepicture” : “/public/client_store/.../21420847.png”,
 “resume” : “/public/client_store/.../2133447.pdf”,
 “gender” : “Male”,
 “demographic” : “Human”,
 “description” : “Biography.”,
 “location” : “Dubai, UAE” ,
 [
   {
      “education_id” : 1,
      “fk_uid” : 100,
      “degree_type” : “Bachelors”,
      “degree_name” : “Computer Science”,
      “university” : “SFSU”,
      “beginDate” : “11-30-2020”,
      “endDate” : null,
      “onGoing” : “1”,
   },
   ...
 ],
[
 {
      “experience_id” : 1,
                “fk_uid” : 100,
                “experience” : “Full-Stack Web Developer”,
                “employer” : “Mozilla, Inc.”,
                “Description” : “Working on FireFox”,
                “beginDate” : “02-30-2020”,
                “endDate” : “03-30-2021”,
                “onGoing” : “0”,
            },
            ...
        ],
        [
          {
                “rating_id” : 1,
                “rating” : 10,
                “rated” : 100,
                “rated” : 101,
                “review” : “Great...”,
                “dateRated” : “02-30-2020”,
                “title” : “Hard Worker”,
            },
          ...
        ],
        [],
    ]
```

b. Code: 200 (teacher profile)

```
[
    "uid" : 100,
    "username" : "ramzi",
    "firstName" : "Ramzi",
    "lastName" : "Abou Chahine",
    "dateOfBirth": "11-30-2020",
    "accType" : "Teacher",
   "currentProfession" : "Computer Science Teacher",
   "profilepicture" : "/public/client_store/.../21420847.png",
   "resume" : "/public/client_store/.../2133447.pdf",
   "gender" : "Male",
   "demographic" : "Human",
   "description" : "Biography.",
   "location" : "Dubai, UAE" ,
   [
     {
        "education_id" : 1,
        "fk_uid" : 100,
        "degree_type" : "Bachelors",
        "degree_name" : "Computer Science",
        "university" : "SFSU",
        "beginDate" : "11-30-2020",
        "endDate" : null,
        "onGoing" : "1",
     },
     ...
   ],
  [
    {
        "experience_id" : 1,
        "fk_uid" : 100,
        "experience" : "Teacher",
        "employer" : "SFSU",
        "Description" : "I teach class Intro to Computer Science",
                “beginDate” : “02-30-2020”,
                “endDate” : null,
                “onGoing” : “1”,
              },
              ...
          ],
          [],
          [],
      ]
```

c. Code: 200 (employer profile)

```
  [
        “uid” : 100,
        “username” : “ramzi”,
        “firstName” : “Ramzi”,
        “lastName” : null,
        “dateOfBirth”: “11-30-2020”,
        “accType” : “Employer”,
       “logo” : “/public/client_store/.../21420847.png”,
       “description” : “Biography.”,
       “location” : “Dubai, UAE” ,
       [],
       [],
       [],
       [
         {
            “listing_id” : 1,
            “jobTitle” : “Junior Front End Developer”,
            “location” : “San Francisco, CA, USA”,
            “expiryDate” : “03-07-2021”,
            “expired” :   0,
            “logo” : “/public/client_store/logos/2133420847.png”,
         },
        ...
       ]
```

]

2. /edit - GET
    - Navigate to edit form to modify profile
3. /search - GET
    - Search for user, employer, or job listing submitting a form,
    - Parameters:
        a. Search Term
        b. Filter
        c. Redirect URL (URL where search was conducted)
    - Response Types: application/json
        a. Code: 200
            - Users Search:

```
[
 “Users”,
 [
  {
   “username” : “johnsmith”,
   “firstName” : “John”,
   “lastName” : “Smith”,
   “verified” : 1,
   “currentProfession” : “Computer Science Student”,
   “profilepicture” : “/public/client_store/.../3213123.jpeg”,
   “location” : “San Mateo, CA, USA”,
   “uscore” : 0.18,
   “pscore” : 0
  },
  ...
 ]
]
```

            - Employers Search:

```
[
  “Employers”,
  [
{
 “username” : “skillseek”,
 “firstName” : “SkillSeek”,
 “verified” : 1,
 “logo” : “/public/client_store/.../343123.jpeg”,
 “location” : “San Mateo, CA, USA”,
 “uscore” : 0.9,
 “pscore” : 0.5321
},
...
]
]
```

- Job Listings Search:

```
[
 “Job Listings”,
 [
  {
   “username” : “skillseek”,
   “firstName” : “SkillSeek”,
   “logo” : “/public/client_store/.../3213123.jpeg”,
   “listing_id” : 2,
   “jobTitle” : “Human Resources Partner”,
   “description” : “Does HR stuff.”,
   “location” : “San Mateo, CA, USA”,
   “expiryDate” : “03/21/2020”,
   “uscore” : 0,
   “jscore” : 0.8
  },
  ...
 ]
]
```

4. /connect/add/{userid} - POST
    - Form connection with another user.
    - Response Types:
        a. Code: 303 (Connection request sent)

5. /connect/delete/{userid} - POST
    - Remove connection between two users.
    - Response Types:
        a. Code: 303 (Connection successfully deleted)
6. /users/logout - POST
    - Terminate user session.
    - Response Types:
        a. Code: 302 (Redirect to homepage after logout)

Students
1. /joblisting/apply/ - GET
    - Apply to job listing
    - Parameters:
        a. Listing ID
        b. Redirect URL
    - Response Types:
        a. Code: 303 (On successful application)
        b. Code: 200 (On unsuccessful application)

Teachers
1. /rate/add/{userid} - POST
    - Add a review and rating on a student's profile.
    - Response Types:
        a. Code: 200 (Invalid input)
        b. Code: 302 (Redirect on successful review)
2. /rate/delete/{userid}/{rating_id} - POST
    - Delete a review and rating on a student's profile.
    - Response Types:
        a. Code: 200 (Invalid input)
        b. Code: 302 (On successful deletion)

Students & Teachers

1. /edit/user - POST
   - Edit request for students and teachers
   - Parameters:
     a. Resume
     b. Profile Picture
     c. First Name
     d. Last Name
     e. Date Of Birth
     f. Gender
     g. Demographics
     h. Description
     i. Location
   - Response Types:
     a. Code: 302 (Redirect on successful edit)
     b. Code: 200 (Invalid input)
     c. Code: 500 (Server error)

2. /education/add - POST
   - Add educational experience to the profile.
   - Parameters:
     a. Degree Name
     b. Degree Type
     c. University
     d. Start Date
     e. End Date
     f. On Going
   - Responses Type:
     a. Code: 304 (Redirect on successful addition)
     b. Code: 200 (Invalid input)
     c. Code: 500 (Server error)

3. /education/delete/{userid}/{education_id} - POST
    - Removes an educational experience from the profile.
    - Responses Type:
        a. Code: 302 (On successful addition)
        b. Code: 200 (Invalid input)
        c. Code: 500 (Server error)

4. /experience/add - POST
    - Add work experience to the profile.
    - Parameters:
        a. Experience
        b. Employer
        c. Description
        d. Start Date
        e. End Date
        f. On Going
    - Responses Type:
        a. Code: 302 (Redirect on successful addition)
        b. Code: 200 (Invalid input)
        c. Code: 500 (Server error)

5. /experience/delete/{userid}/{experience_id} - POST
    - Remove work experience from the profile.
    - Responses Type:
        a. Code: 303 (Successful removal)
        b. Code: 303 (Failed removal)
        c. Code: 500 (Server error)

6. /connect/accept/ - GET
    - Accept connection request.
    - Parameters:
        a. Connection ID
    - Response Type:
        a. Code: 303 (Connection successfully established)

7. /connect/deny/ - GET
    - Refuse connection request.

- Parameters:
    a. Connection ID
- Response Type:
    a. Code: 303 (Connection successfully refused)

Employers

1. /joblisting/{username}/{listing_id} - GET
    - See posted job listing information and applicants.
    - Response Types: application/json
        a. Code: 200
        ```
        [
          {
           “username” : “janedoe”,
           “firstName” : “jane”,
           “lastName” : “doe”,
           “profilepicture” : “/public/client_store/.../1234523.jpeg”,
           “location” : “San Jose, CA, USA”
          },
          ...
        ]
        ```
2. /edit/org - POST
    - Edit request for employers
    - Parameters:
        a. Logo
        b. First Name (organization name)
        c. Date Of Birth (date established)
        d. Description
        e. Location
    - Response Types:
        a. Code: 302 (Redirect on successful edit)
        b. Code: 200 (Invalid input)
        c. Code: 500 (Server error)
3. /joblisting/add - POST
    - Add a job listing.
    - Parameters:
        a. User ID
        b. Job Title
        c. Description
        d. Location

    e. Expiry Date
- Response Types:
    a. Code: 200 (Invalid input)
    b. Code: 304 (Redirect on successful job listing post)

4. /joblisting/delete/{listing_id} - POST
    - Delete a job listing.
    - Response Types:
        a. Code: 200 (Permission denied)
        b. Code: 303 (Successful deletion)

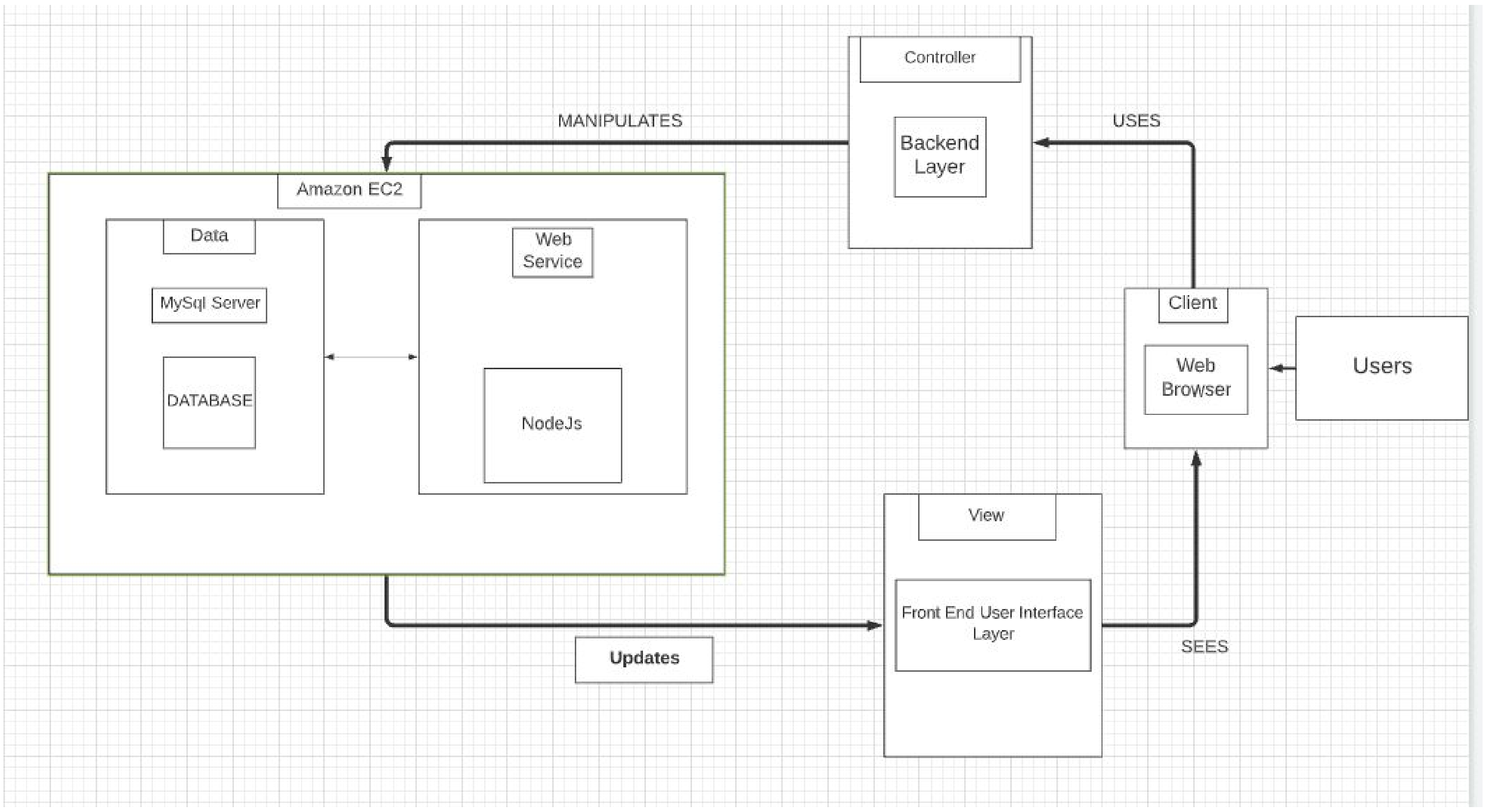
5. /joblisting/hide/{listing_id} - POST
    - Hide a job listing.
    - Response Types:
        a. Code: 200 (Permission denied)
        b. Code: 303 (Successfully hidden)

Server

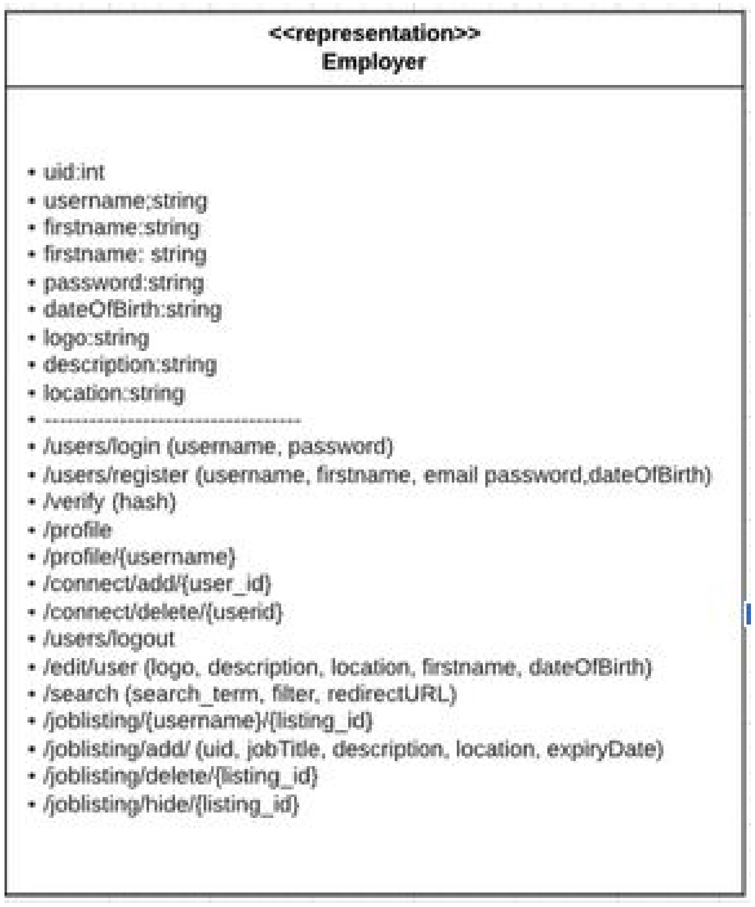
1. /verify/initialize - POST
    - Called by the server to initialize the verification hash and send email to the user.
    - Parameters:
        a. First name
        b. User ID
        c. Email
    - Responses Type:
        a. Code: 200 (Verification email could not be sent)
        b. Code: 200 (Initialization request complete)
        c. Code: 500 (Server Error)

High-Level UML Diagrams

Deployment Diagram

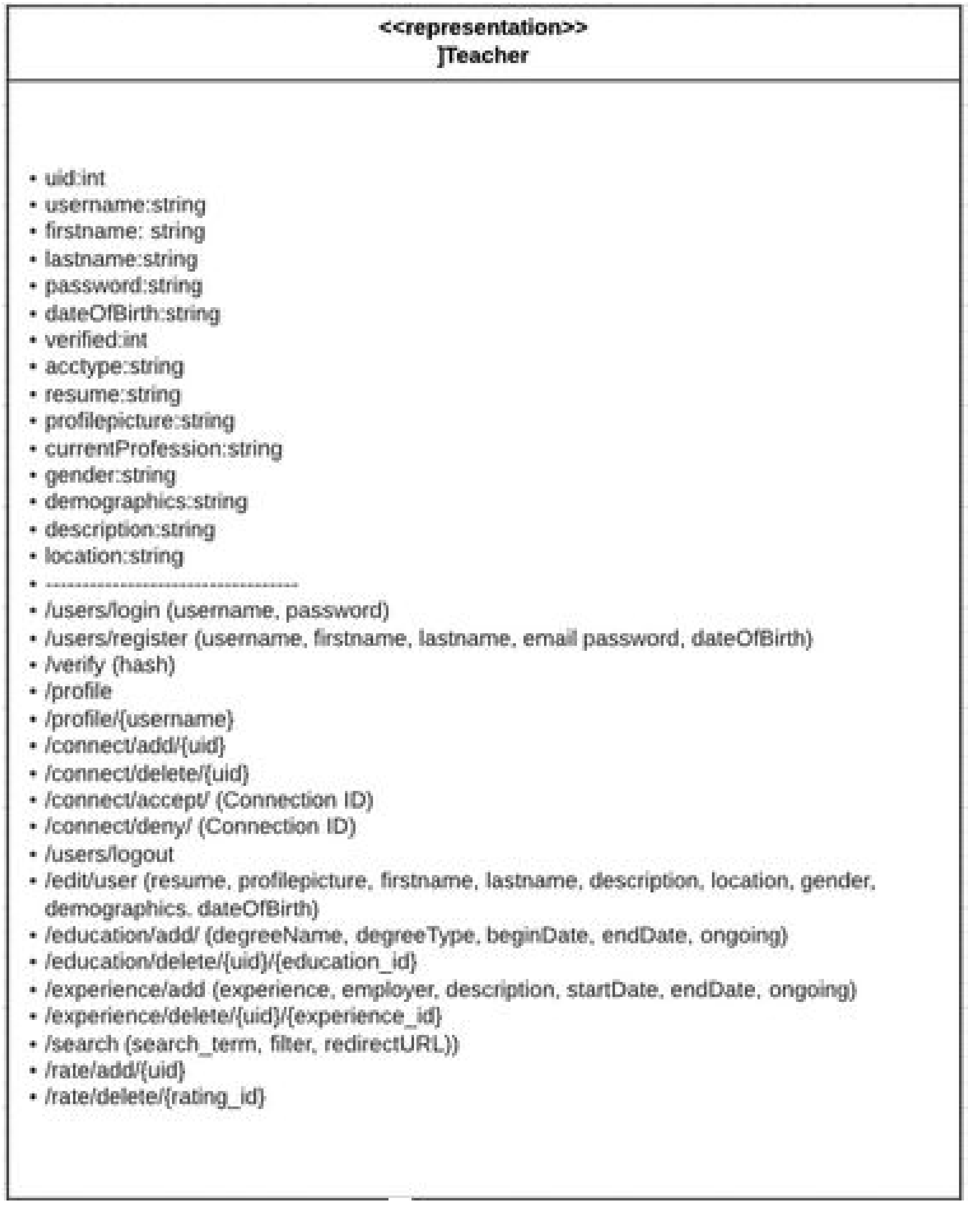


Class Diagrams



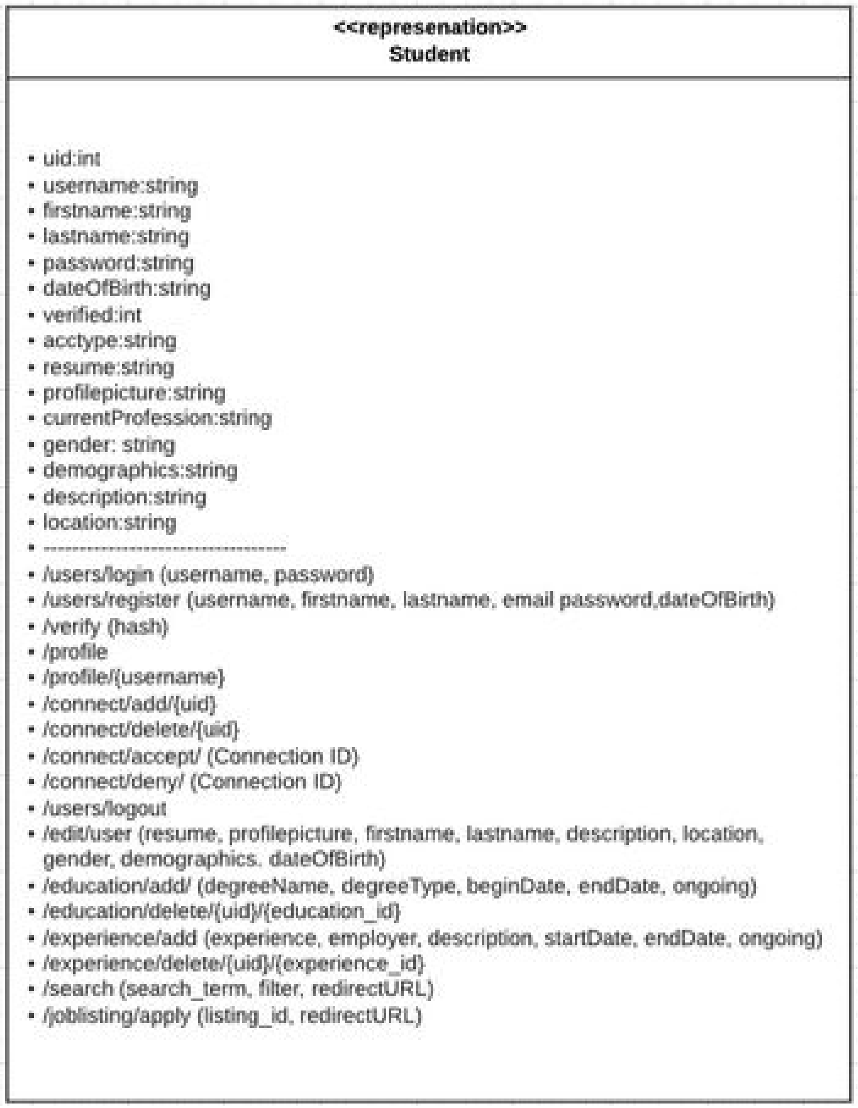
<<representation>>
Employer

- uid:int
- username;string
- firstname:string
- firstname: string
- password:string
- dateOfBirth:string
- logo:string
- description:string
- location:string
- -----------------------------------
- /users/login (username, password)
- /users/register (username, firstname, email password,dateOfBirth)
- /verify (hash)
- /profile
- /profile/{username}
- /connect/add/{user_id}
- /connect/delete/{userid}
- /users/logout
- /edit/user (logo, description, location, firstname, dateOfBirth)
- /search (search_term, filter, redirectURL)
- /joblisting/{username}/{listing_id}
- /joblisting/add/ (uid, jobTitle, description, location, expiryDate)
- /joblisting/delete/{listing_id}
- /joblisting/hide/{listing_id}



<<representation>>
]Teacher

- uid:int
- username:string
- firstname: string
- lastname:string
- password:string
- dateOfBirth:string
- verified:int
- acctype:string
- resume:string
- profilepicture:string
- currentProfession:string
- gender:string
- demographics:string
- description:string
- location:string
- ---------------------------------
- /users/login (username, password)
- /users/register (username, firstname, lastname, email password, dateOfBirth)
- /verify (hash)
- /profile
- /profile/{username}
- /connect/add/{uid}
- /connect/delete/{uid}
- /connect/accept/ (Connection ID)
- /connect/deny/ (Connection ID)
- /users/logout
- /edit/user (resume, profilepicture, firstname, lastname, description, location, gender, demographics. dateOfBirth)
- /education/add/ (degreeName, degreeType, beginDate, endDate, ongoing)
- /education/delete/{uid}/{education_id}
- /experience/add (experience, employer, description, startDate, endDate, ongoing)
- /experience/delete/{uid}/{experience_id}
- /search (search_term, filter, redirectURL))
- /rate/add/{uid}
- /rate/delete/{rating_id}



<<represenation>>
Student

- uid:int
- username:string
- firstname:string
- lastname:string
- password:string
- dateOfBirth:string
- verified:int
- acctype:string
- resume:string
- profilepicture:string
- currentProfession:string
- gender: string
- demographics:string
- description:string
- location:string
- ----------------------------------
- /users/login (username, password)
- /users/register (username, firstname, lastname, email password,dateOfBirth)
- /verify (hash)
- /profile
- /profile/{username}
- /connect/add/{uid}
- /connect/delete/{uid}
- /connect/accept/ (Connection ID)
- /connect/deny/ (Connection ID)
- /users/logout
- /edit/user (resume, profilepicture, firstname, lastname, description, location, gender, demographics, dateOfBirth)
- /education/add/ (degreeName, degreeType, beginDate, endDate, ongoing)
- /education/delete/{uid}/{education_id}
- /experience/add (experience, employer, description, startDate, endDate, ongoing)
- /experience/delete/{uid}/{experience_id}
- /search (search_term, filter, redirectURL)
- /joblisting/apply (listing_id, redirectURL)

Component Diagram

Due to the size of the diagram, we were forced to host it externally.
It can be found here. (imgur.com)

The Actual Key Risks For The Project

- Skill Risks

  Our team is very diverse in our skills. There are different advantages and disadvantages to having so many different skill levels in one group. A disadvantage would be that it was hard to understand all the given information at once. If all the team members had equal levels of skills, it would be easier to do work efficiently. However, we all have different levels of skills, and sometimes it's burdensome for the specific work where one of the teams is perfect for that part and ones not. Each of us had our own experiences regarding our work. The main point is to concentrate on solving issues and using each person's strengths. We can solve these issues by modularizing the tasks and better understanding the capabilities of each member. If a member should encounter difficulties completing their tasks, then he/she should be in contact with their teammates/leader as soon as possible. If we do this then we can work together to solve that specific part.

- Schedule risks

  The main risk for scheduling a meeting is finding a time when everyone can meet for an hour or more. Our lives outside of class seem to keep us all busy. Sometimes a person had to work and could not attend the meeting, they would enter the meeting late or leave early. We managed to discuss our work after class or before class. Not attending meetings would lead to confusion about the tasks requiring completion and how fulfilled tasks were met/implemented, which may slow the team's pace. We could minimize this risk by giving each other brief information on class day or upon task completion. If anyone else had questions, they could ask and start working on helping each other. Because so many people have different obligations, and we tried our best not to leave work for the last moment because this only leads to many more problems. We are aware of the problems that we could face and are now working together much better.

- Technical Risks

  There are always technical risks associated with building applications, especially those that require user information to function. The presence of potential vulnerabilities in the program could lead to exploitation by bad actors for several reasons, including stealing user information. Safe-guarding from such attacks requires an in-depth understanding of how to secure online applications from attacks like SQL injections. Another technical risk that may arise is coping with traffic when the application becomes public and operational. Ensuring a satisfactory user experience requires the website must remain up and operate smoothly at all times, lest users become frustrated with the application's performance and resort to using our competitors. Meeting those criteria necessitates that the application's design is scalable and efficient, which may entail using a more powerful server or several. Developers may encounter technical risk when implementing an application based upon the high-level description agreed upon, as there is always a potential that the method chosen proves incompatible. For example, a part of the application implemented by another team member may not be compatible with the approach adopted in a different part of the application; this would require changes that could prove to be extremely time-consuming.

- Teamwork Risks

  This ties in with skill risks. Some of the team members were willing to attend meetings and work together, but they faced issues such as family problems. However, mostly, the problem was time management. Due to covid, we have team members in different time zones. To minimize the time problem, we do our meetings accordingly, so that almost everyone is available to collaborate. This makes it easier for each person to know and do their part of the work.

- Legal/Content Risks

  Risks here can include disputes and regulations. Legal risks of dispute can be a legal claim is made such as employee misconduct, accidents, etc. while on the other hand, in regulatory, there is a high chance that we might lose our site because it will get turned down by the companies, and the reason will adversely impact the economic value. It may get subjected to regulations from government institutions or agencies. Another could be where we will be aware of taking care of sensitive/inappropriate images or copyright material being uploaded to websites. To take care of this, it's not hard to take care of having high security which is not going to completely prevent but will reduce the users from uploading those kinds of stuff on the website. We can use the report/flag button where if users see inappropriate material, then the user can report it. Another, we can use guidelines that also warn users not to upload such material online which causes them to have their account removed, and also the company may face legal charges for allowing or having illegal information on the website Also, we always have to take care of what is being uploaded or written on the website. Since we So, to avoid these high risks in the future, we always have plans ready, such as we discuss above and since we are using AWS for our website, we can use " Content moderation " which detects inappropriate, unwanted, or offensive material which is uploaded on the website and helps them removed from the website. In this way, we can cut the amount of legal risk, copyright material, inappropriate images, or PDF being uploaded illegally and continue to operate normally.

Project management

From the beginning of the development process, we have used Discord to communicate. We all have our parts to work on, but if anyone needs help on their part, we will be helping them. Also, we do meetings on Discord either before class, after class, or on Friday mornings. Other than these days, we are active on Discord to check on each other's work. Whenever we start a new milestone, we first break it down and explain the parts of the project in detail to each team member. Team members decide which one they feel comfortable with and can work on. Also, we have decided to use Trello to organize tasks and keep track of members' assigned tasks.

Milestone 1 Modifications

Personas and Main Use Cases

1. Students:

Our main and most numerous users shall be students as they are the primary focus of our application. As upcoming and fresh graduates, students are motivated to find a job and begin their careers to be able to sustain themselves. They tend to be creative, optimistic, and very eager to prove themselves. As SkillSeek users, they would be responsible for maintaining their profile and keeping their information and resume up to date to attract employers' interest. They are also encouraged to persuade their instructors to review them in order to differentiate themselves and increase their probability of being hired. A pain point for any student user is that they will have to compete with other students when applying for jobs and that may dissuade, frustrate, or possibly deter them from using our service. Another frustration that students will likely face is the difficulty in finding employment due to the saturation of the job market or their education and experiential background. A student's ultimate goal for using SkillSeek is finding a well paying job suitable to their interests and personality.

Examples of potential student users:

- Grace, a high school graduate, is looking for a job. She is really passionate about becoming a Barista. With her friend's recommendation, she goes to Skillseek and decides to create an account. Grace eventually decides to use SkillSeek's search functionality to look for a job in her desired area. She comes up with 20 job listings matching her criteria and proceeds to apply to them directly through the platform.

- Bob, a Computer Science student at a US university, is looking for internships while completing his degree to accumulate work experience and build a compelling resume. He creates a SkillSeek account, customizes his profile, and uploads his resume. He begins to seek out his preferred organizations to view available internships opportunities. He applies to their applicable listings and awaits a response from potential employers.
- Emelia, a recent graduate specializing in Environmental Science, found a job through SkillSeek at a waste management company but is unsatisfied with her wage and the work environment. She decides to look for different opportunities, so she routinely returns to SkillSeek to check for the latest job listings matching her criteria and applying to those that suit her. She is contacted by several employers who offer her positions at their organization and compares the offers. She settles on an environmentally focused non profit and proceeds to submit her resignation in pursuit of an exciting new opportunity.

Use Cases:

1. **Finding a Job:**

A student is looking for a job so they decide to join SkillSeek to seek out opportunities.

1. The student registers for an account with SkillSeek.
2. The student checks his email to complete verification email to verify his account and complete registration.

3. The student logs in to their account and edits their profile by including some brief information and uploading a profile picture and resume.
4. The student utilizes search to look for jobs that they are interested in and then hits the apply button.

2. Finding Friends/Networking:

Students are looking to find friends / new networks on SkillSeek.com.

1. The student first accesses their account.
2. The student then hits friends request button and types in the friends name
3. The student waits for the friend to accept or reject the friend request.
4. Once mutuals, both parties will be able to private message each other and network.
5. The student will receive a notification (if enabled) and an email if and when they are on the receiving end of a friend request where they will be able to accept or deny.

3. Finding a company

Students are looking for whether a certain company has any job listings on SkillSeek.com.

1. The student first accesses their account.
2. The student utilises the search functionality to input their company of interest.
3. The student will be able to see any job listings the company of interest has available and displayed on their page.
4. If the student is interested in any listing, they are able to click the apply button.

2. Teachers:

Our least common users will most likely be teachers; however, they are vital members of the community as they are responsible for recommending students to employers on our platform. The teacher's motivation is to help their student find jobs and they can do so by simply giving them review. Since we are aware that teachers across the world are extremely preoccupied with their families and work we know that this results in teachers having a limited time for writing recommendations, thus our platform's review system would be a much faster alternative option to give a review to students than the typical recommendation letter you would receive from your professor. Also, at any given moment of their career, a teacher could use our platform to apply for a job similar to a regular student if they wish to do so .

Example of potential teacher users:

- Mr. Johnson, a political science teacher at San Francisco State University, wants to review one of his students. Unfortunately, he doesn't really know a platform on the internet that has this certain feature. So Mr.Johnson was recommended to go on SkillSeek by one of his colleagues who uses SkillSeek routinely to give ratings to his students. Consequently, Mr. Johnson then decided to go to Skillseek and make a teacher account . Once, Mr. Johnson completed verifying his account and editing his profile, he proceeded to visit a student's profile to rate them and he does this simply by looking up the student name or unique username in the search bar. Once Mr. Johnson verifies that it's the right student, he then proceeds to write a review on his student's account.

Use Cases:

1. **Finding/Reviewing the student:**

Teachers are looking for a platform to review their students. They came across Skillseek.com where they can easily connect with students and grade them.

1. A teacher registers for a new teacher account.
2. The teacher proceeds to verify their account.
3. The teacher may look up their student's name in the search bar.
4. The teacher can then review their student's profile.

3. Employers:

The last target audience for SkillSeek are employers; they play an imperative role at SkillSeek which is providing students and teachers with jobs. Employers have a clear and direct incentive to use our platform as it provides them with an effective and efficient way of sourcing up and coming talent. Employers can also benefit from the exposure our platform can provide as it can serve as a form of marketing. One of the main frustrations employers have to cope with, in terms of recruitment, is the sheer volume of applications they receive and the difficulty of finding suitable applicants effectively. To cope with these difficulties, organizations have resorted to outsourcing the process to a third party and even hiring large human resources departments. By using the straightforward user interface our platform offers, employers can inspect applicant profiles with the click of a button and view all the information that they require to make their decision.

Examples of potential Employer users:

- A tech-company looking for graduate students to fill vacant job positions. The company decides to use SkillSeek and proceeds to create an account and post a job listing for students to apply for. They come up with more than 50 applicants for their organization and are able to review their profiles directly to ensure they meet the desired criteria. They proceed to schedule interviews with eligible candidates until they are able to find the right matches for the open positions.

Use Cases:

1. **Looking for Employees:**

   An organization is looking for talent for their company, they came across Skillseek.com where they can find new people for their company.

   1. An employer signs up for a new account.
   2. The employer then proceeds to create their first job listing by providing information about the position they are trying to fill.
   3. The employer waits for candidates to apply for the job listing.
   4. Once candidates have applied, the employer will begin to receive emails to notify them.
   5. The employer logins into SkillSeek and navigates to their profile to view posted job listings.
   6. The employer selects the job listing that they were notified about to view the applicants.
   7. The employer may then select each applicant to navigate to their profile and view their experiences and resume.
   8. The employer then emails the applied candidate to schedule a meeting for an interview.

9. If the employer is satisfied and the job vacancy has been filled the employer may choose to hide or delete the job listing.
10. If the vacancy has not been filled, the employer may continue seeking viable candidates.
11. If the job listing expires, the employer can continue to examine applicant profiles for a period of two months before the job listing and its applicants are cleared.

High-level System Architecture and Technologies Utilized

Listed below are software technologies that our team has agreed upon. These may be subject to slight changes in the future, but will stay mostly consistent to provide a solid experience for the end-user.

High-Level System Architecture: Model View Controller Architecture

Deployment Model: Software as a Service (SaaS)
Server-side Technologies:

- Host - AWS EC2 t2.micro (1x vCPU, 1gb RAM)
- OS - Ubuntu Linux 20.04 (LTS)
- Web Server - NodeJS 14.15.5 (LTS)
- Reverse Proxy and Static File Server - NGINX/1.18.0 (Ubuntu)
- Database - MySQL 8.0.21
- Storage Engine - InnoDB
- Web Analytics Service - Google Analytics
- Deployment Platform - AWS CodeDeploy (may switch to AWS CodePipeline)

Front End Frameworks / Additional Technologies:

- Framework: ExpressJS 4.17.1
- Template Engine: HandleBars 4.7.7
- Markup Language: HTML5
- Styling Language: CSS3

Tools:

- Virtualization Software - VirtualBox & VMware
- Database Design Tool - MySQL Workbench
- Versioning System - Git
- Repository Host - GitHub

Targeted Browsers:

- Google Chrome
- Mozilla Firefox