Final Report

United States Colleges and University
Faculty and Student Database

Group 12

Members:
Mauzor Ilonzo
Steven Smail
Mark Yo

Overview

The purpose of our desktop app is to develop a system where faculty and students that have attended any university/college in the United States may log the institutions they have taught or studied at respectively. They will be able to access their own information, as well as look up the specifics about each school in the country. If they are a professor, they can access student data. We got our data for the Colleges from the following link:
https://public.tableau.com/s/sites/default/files/media/Resources/IPEDS_data.xlsx

Implementation Options

In our first report we suggested to using a console for text application, Electron for a desktop app, and a mixture of Python and SQLAlchemy for implementing a website. We ended up using Electron to implement our database system. Since Electron utilizes Node.js, we were able to Node.js' MySQL module to implement our queries.

We utilized the quickstart code Electron provides to jumpstart the creation of our Electron-based project. We also utilized the instructional information provided by w3schools.com as a guide for using Node.js and the Node.js MySQL module. Please see below for links.

Electron Quick Start: https://github.com/electron/electron-quick-start
w3schools.com Node.js: https://www.w3schools.com/nodejs/default.asp

Refer to the README for information on how to run the program.

Database Tables

Our database system is comprised of the following 5 tables, Colleges, Students, Professors, Attends & Logging. We had to add the loggings table in order to keep track of the users logged into the database.

Colleges(cid: INTEGER, name: STRING, zip: INTEGER, hdeg: STRING, county: STRING, appnum: INTEGER, adnum: INTEGER, tuition: INTEGER, state: STRING, tenum: INTEGER, ugenum: INTEGER, genum: INTEGER)

- **Primary key:** cid
- **Foreign key:** None

Each entry in the Colleges table contains all of the information for a given college in the database. When a user searches for information about colleges, the information will be queried from this table. Each college has a college ID number, college name, highest degree offered, county name, total number of applicants, total number of admissions, cost of tuition and fees, state, zip code, total enrollment, undergraduate enrollment, and graduate enrollment.

Students(sid: INTEGER, fname: STRING, lname: STRING, dob: DATE, state: STRING, zip: INTEGER, user: STRING, pass: STRING)

- **Primary key:** sid
- **Foreign key:** None

Each entry in the Students table refers to a specific student-user who attends any of the universities in the database. Each student entry contains their student ID number, first name, last name, date of birth, state & zip code, and their username and password for the website. Information from this table will be used for corresponding queries from professor-users as well as the authentication of student-users attempting to login.

Professors(pid: INTEGER, fname: STRING, lname: STRING, dob: DATE, state: STRING, zip: INTEGER, salary: FLOAT, user: STRING, pass: STRING)

- **Primary key:** pid
- **Foreign key:** None

Each user in the Professors table refers to a specific professor who has taught in any of the universities in the database. Each professor's entry would contain their professor ID number, first name, last name, date of birth, state & zip code, salary and their username and password for the website.
Similar to the Students table, the authentication of professor-users and the search queries by student-users also use information from this table.

Attends(aid: INTEGER, pid: INTEGER, sid: INTEGER, cid: INTEGER, dstart: DATE, dend: DATE)

- **Primary key:** aid
- **Foreign key:** pid(Professors.pid), sid(Students.sid), cid(Colleges.cid)

Every entry in the Attends table documents when either a student or a professor attends a given university. In this table, if an entry has a pid then the sid value would be set to NULL and if an entry has an sid, then the pid would be set to NULL. We created a separate ID (aid) to use as a primary key because if we just used the pid, sid, and cid as the primary key it would be difficult to keep track of students and professors who attended a single university more than once.

Logging(lid: INTEGER, user: STRING, uid: INTEGER)

- **Primary key:** lid
- **Foreign key:** uid(Professors.pid or Students.sid), user(Professors.user or Students.user)

Every entry in the Logging table keeps track of all of the users that are logged into the system. Since either a student or a professor can be logged in, the uid foreign key determines whether the user in question is a student or a professor.

SQL Table Creation

This section contains the code we used in order to populate the tables with data, please refer to the *.sql files that are paired with this report for the complete sample data that is also populated alongside these tables.

Colleges Table:

```
DROP TABLE IF EXISTS colleges;
CREATE TABLE colleges(
   cid     INTEGER  NOT NULL PRIMARY KEY
  ,name    VARCHAR(200) NOT NULL
  ,zip     INTEGER  NOT NULL
  ,hdeg    VARCHAR(200) NOT NULL
  ,county  VARCHAR(200) NOT NULL
  ,appnum  INTEGER
  ,adnum   INTEGER
  ,tuition INTEGER
  ,state   VARCHAR(20) NOT NULL
  ,tenum   INTEGER
  ,ugenum  INTEGER
  ,genum   INTEGER
);
```

Professors Table:

```
DROP TABLE IF EXISTS professors;
CREATE TABLE professors (
    pid INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
    fname VARCHAR(20) NOT NULL,
    lname VARCHAR(20) NOT NULL,
    dob DATE NOT NULL,
    state VARCHAR(20),
```

```
   zip INTEGER,
   salary FLOAT(200,2),
   user VARCHAR(20) NOT NULL,
   pass VARCHAR(20) NOT NULL
);
ALTER TABLE professors AUTO_INCREMENT=5001;
```

Students Table:

```
DROP TABLE IF EXISTS students;
CREATE TABLE students (
   sid INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
   fname VARCHAR(20) NOT NULL,
   lname VARCHAR(20) NOT NULL,
   dob DATE NOT NULL,
   state VARCHAR(20),
   zip INTEGER,
   user VARCHAR(20) NOT NULL,
   pass VARCHAR(20) NOT NULL
);
```

Attends Table:

```
DROP TABLE IF EXISTS attends;
CREATE TABLE attends (
   aid INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
   uid INTEGER NOT NULL,
   cid INTEGER NOT NULL,
   dstart DATE NOT NULL,
   dend DATE NOT NULL
);
```

Logging Table:

```
DROP TABLE IF EXISTS logging;
CREATE TABLE logging (
   lid INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
   user VARCHAR(20) NOT NULL,
   uid INTEGER NOT NULL
);
```

Queries

Query 1.a:

// Perform a query to check the status of the most recently logged in user.

```
$query =        "SELECT *
                FROM logging
                ORDER BY lid DESC"
```

Query 1.b:

// Perform a query that returns all of the attributes of every entry in the logging table. This query checks how many people are logged in.

```
$query =        "SELECT *
                FROM logging"
```

Query 2:

// Performs a query that returns the names of every available college in alphabetical order.

```
$query1 =       "SELECT name
                FROM colleges
                ORDER BY name ASC"
```

Query 3:

// Query that returns the name and the range of attendance of the currently logged in user.

```
$query =        "SELECT c.name, a.dstart, a.dend
                FROM attends a, colleges c
                WHERE a.uid = ? AND a.cid = c.cid GROUP BY c.name"
```

Query 4:

// Query that returns the cid from college name and then inserts a new entry in the attends table

```
$query =        "SELECT cid
                FROM colleges
                WHERE name = ?"
$query2 =       'INSERT INTO attends (uid, cid, dstart, dend) VALUES (?, ?, ?, ?)'"
```

Query 5.a:

// Query to check if the user info is tied to a professor

```
$query =        "SELECT *
                FROM professors
                WHERE user = ? AND pass = ?"
```

Query 5.b:

```
// Query to check if the user info is tied to a student.
$query =        "SELECT *
                 FROM students
                 WHERE user = ? AND pass = ?"
```

Query 6:

```
// Inserts user into the log through a query
$query1 =       "INSERT INTO logging (user, uid)
                 VALUES (?, ?)"
```

Query 7:

```
// Query to check if the user is logged in.
$query =        "SELECT *
                 FROM logging
                 WHERE user = ?"
```

Query 8:

```
// Checks to run the student or professor query. Then registers the new user as a student or
professor, respectively
$query =        "INSERT INTO students (fname, lname, dob, state, zip, user, pass)
                 VALUES (?, ?, ?, ?, ?, ?, ?)"

$query2 =       "SELECT sid
                 FROM students
                 ORDER BY sid DESC LIMIT 1"

$query =        "INSERT INTO professors (fname, lname, dob, state, zip, salary, user, pass)
                 VALUES (?, ?, ?, ?, ?, ?, ?, ?)"

$query2 =       "SELECT pid
                 FROM professors
                 ORDER BY pid DESC LIMIT 1"
```

Query 9:

```
// Query to delete the entries from the logging table's current session
$query =        "DELETE FROM logging"
```

Query 10:

```
//Query of attendance of a single student at any schools
$query =        "SELECT c.name, a.dstart, a.dend
                FROM attends a, students s, colleges c
                WHERE s.fname = ? AND s.lname = ? AND s.sid = a.uid AND a.cid = c.cid
                GROUP BY c.name"
```

Query 11:

```
// Perform a query to check the status of the most recently logged in user.
$query =        "SELECT *
                FROM logging
                ORDER BY lid DESC"
```

Query 12:

```
// Perform a query to print the colleges based on search.
$query =        "SELECT name, hdeg, state, county, zip
                FROM colleges
                WHERE (name LIKE ?) OR (hdeg LIKE ?) OR (state LIKE ?) OR (county LIKE
                ?) OR (zip LIKE ?)
                GROUP BY name"
```

Query 13:

```
// Perform a query to print the colleges based on tuition.
$query =        "SELECT name, tuition, state, county, zip
                FROM colleges
                WHERE tuition < ? GROUP BY name"
```

Query 14:

```
// Perform a query to print the colleges based on acceptance rate
$query =        "SELECT name, (adnum/appnum * 100) AS accepted, state, county, zip
                FROM colleges
                WHERE (adnum/appnum * 100) <= ? ORDER BY (adnum/appnum * 100)
                DESC"
```

Client Functionality

In order to access every feature of our website, each user must register and create an account. Users may choose to either register as a student or as a professor. Once users have created their own account they will then be able to view five different types of pages. These pages include:

Page I: Login

The Login Page allows you to log in to the website and begin your searches. It provides entry for your username and password. If it is your first time logging in to the program, you must register a new account. For the creation of a new account, you will click the "Register" button, enter your information and whether you are a student or professor. You will be logged in immediately after you register automatically. Upon successful login or registration, you will see a "Success" message in green text.

Figure 1 - Registration Page

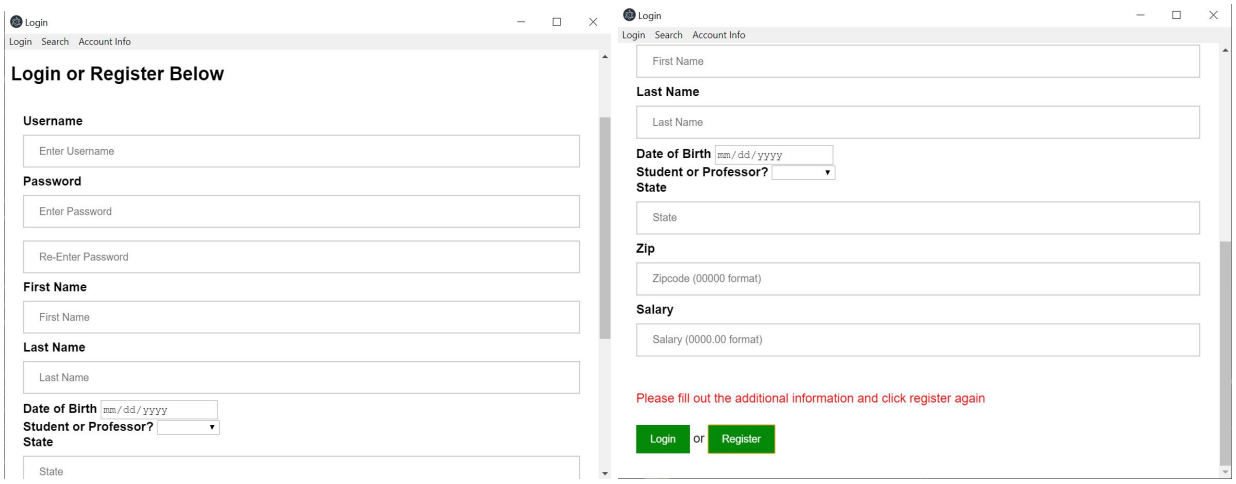
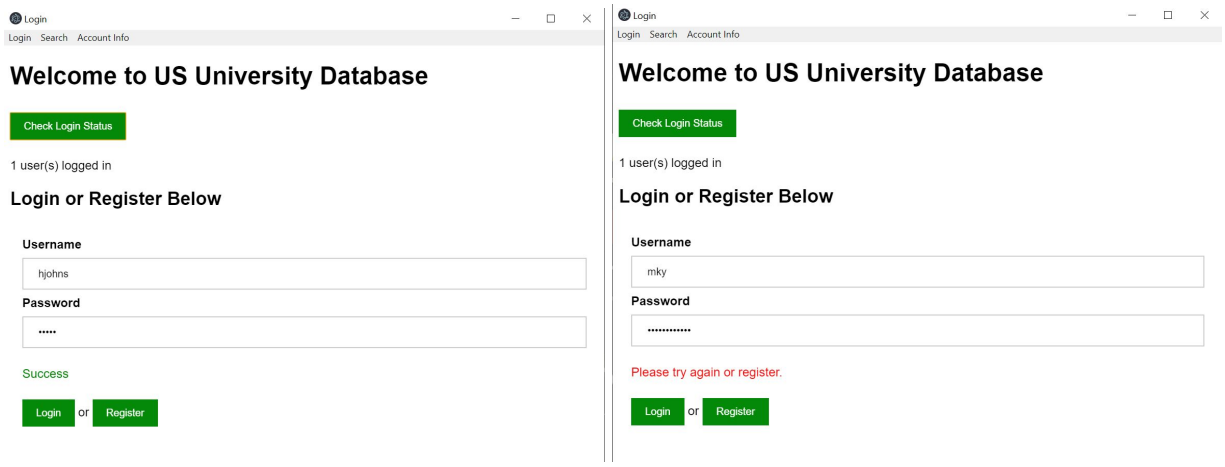


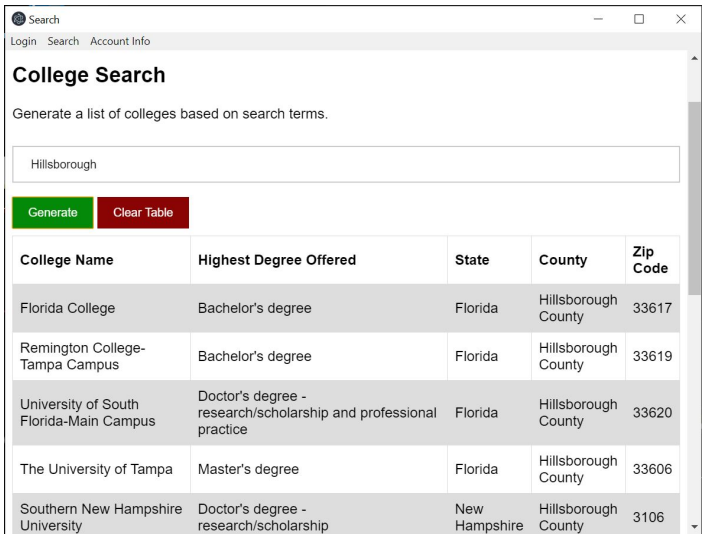
Figure 2 - Login Page with Login Attempts



Page II: Search

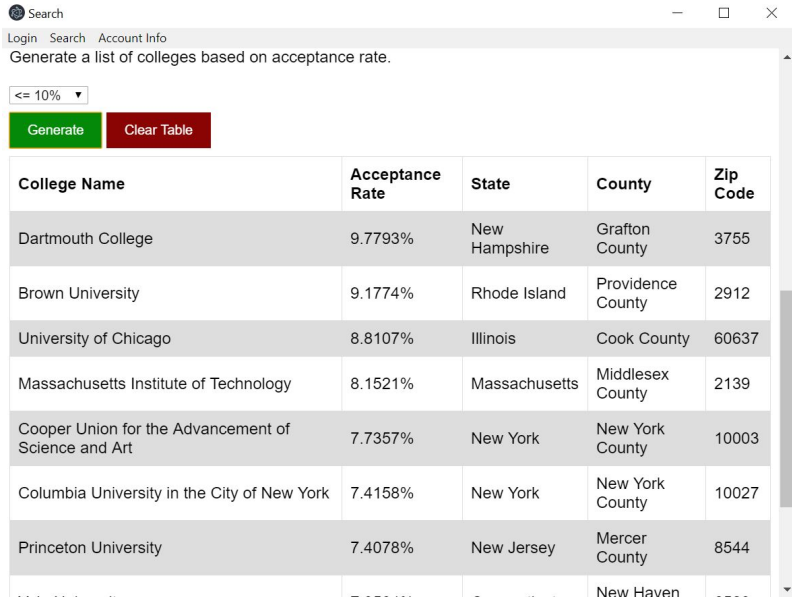
The Search Page allows users to search colleges through the following means: a general search, search by tuition, and search by acceptance rate. Upon completion of the search, the user will click the "Generate" button and it will print a table of relevant results. The "Clear Table" button clears entries in the table for new searches.

Figure 3 - Search Page with Keyword Example



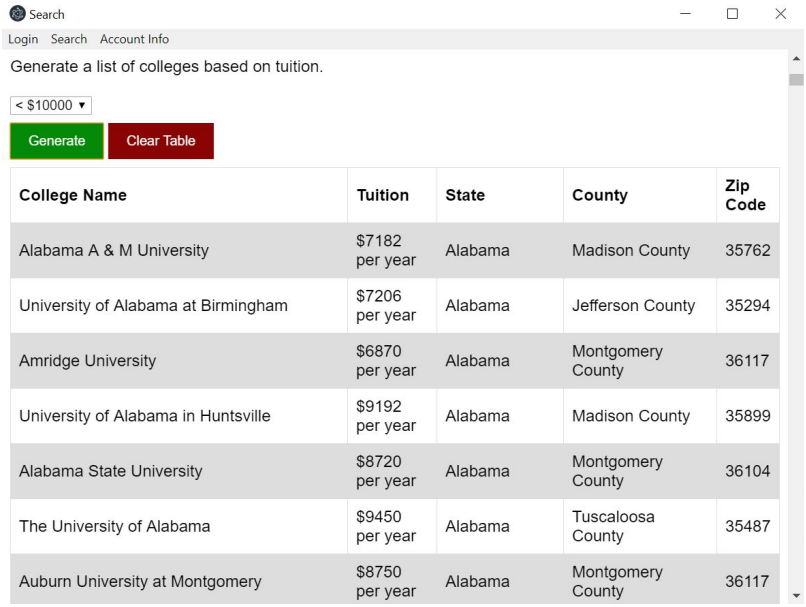
College Name	Highest Degree Offered	State	County	Zip Code
Florida College	Bachelor's degree	Florida	Hillsborough County	33617
Remington College-Tampa Campus	Bachelor's degree	Florida	Hillsborough County	33619
University of South Florida-Main Campus	Doctor's degree - research/scholarship and professional practice	Florida	Hillsborough County	33620
The University of Tampa	Master's degree	Florida	Hillsborough County	33606
Southern New Hampshire University	Doctor's degree - research/scholarship	New Hampshire	Hillsborough County	3106

Figure 4 - Search Page with Acceptance Rate Example



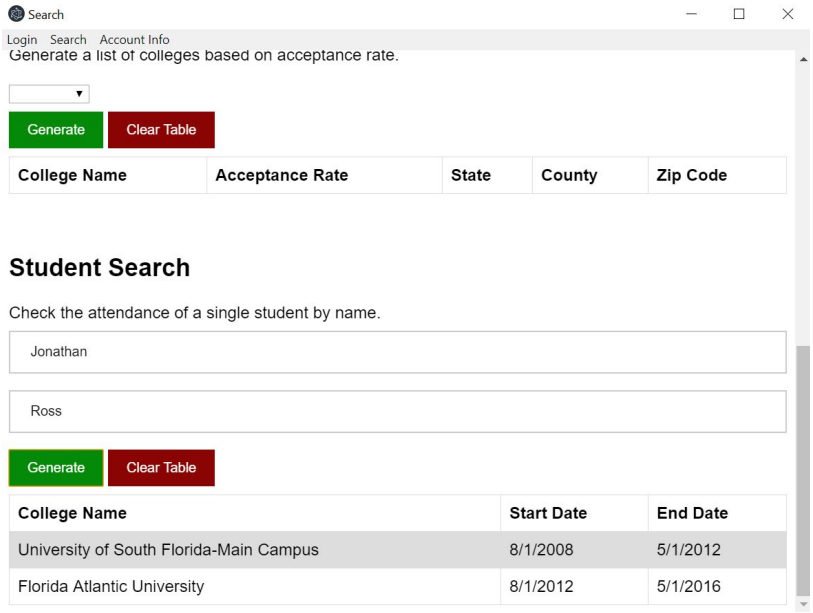
College Name	Acceptance Rate	State	County	Zip Code
Dartmouth College	9.7793%	New Hampshire	Grafton County	3755
Brown University	9.1774%	Rhode Island	Providence County	2912
University of Chicago	8.8107%	Illinois	Cook County	60637
Massachusetts Institute of Technology	8.1521%	Massachusetts	Middlesex County	2139
Cooper Union for the Advancement of Science and Art	7.7357%	New York	New York County	10003
Columbia University in the City of New York	7.4158%	New York	New York County	10027
Princeton University	7.4078%	New Jersey	Mercer County	8544
			New Haven	

Figure 5 - Search Page with Tuition Example



College Name	Tuition	State	County	Zip Code
Alabama A & M University	$7182 per year	Alabama	Madison County	35762
University of Alabama at Birmingham	$7206 per year	Alabama	Jefferson County	35294
Amridge University	$6870 per year	Alabama	Montgomery County	36117
University of Alabama in Huntsville	$9192 per year	Alabama	Madison County	35899
Alabama State University	$8720 per year	Alabama	Montgomery County	36104
The University of Alabama	$9450 per year	Alabama	Tuscaloosa County	35487
Auburn University at Montgomery	$8750 per year	Alabama	Montgomery County	36117

Figure 6 - Search Page with Student Example (Only accessible as a Professor)



College Name	Acceptance Rate	State	County	Zip Code

College Name	Start Date	End Date
University of South Florida-Main Campus	8/1/2008	5/1/2012
Florida Atlantic University	8/1/2012	5/1/2016

Page III: Account Info

Once an account is created, registered users have the option to view and alter their personal attendance information in the Account Info Page. They can generate tables for their attendance information, and add information to their attendance.

Figure 7 - Account Page with the user's attendance

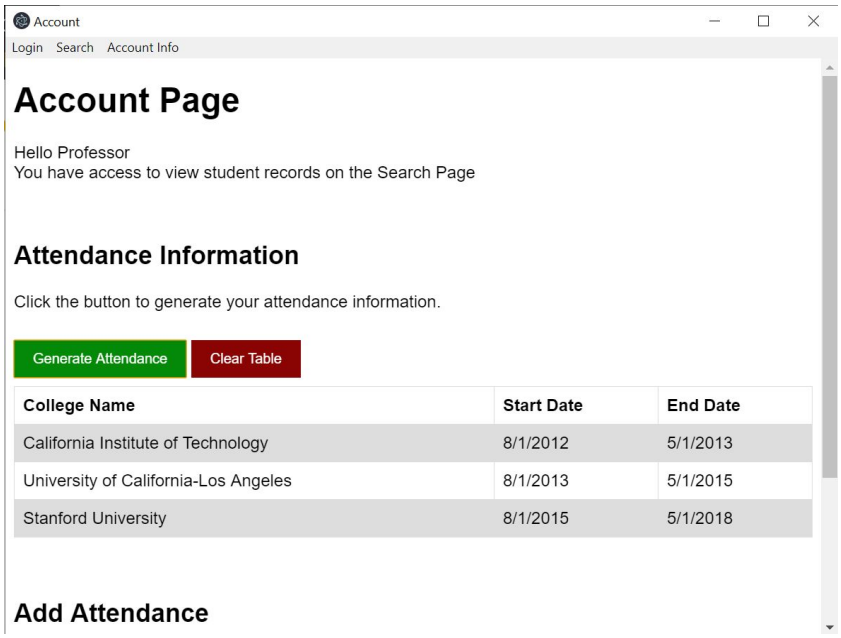


Figure 8 - Adding an Attendance entry on the user's account page