INFSCI 2710 - Database Management - Final Project Report

Neha Chanu (KNC36@pitt.edu)
Quinton Steele (QJS2@pitt.edu)
Abdulelah Abuabat (AAA185@pitt.edu)

April 18th

1 Introduction

The idea, conference gate, overall is a conference registration gate which acts as a hub for conferences across disciplines. Typically users seeking in-field conferences have to rely on Google search, social media, recommendations from friends and family, and in-house advertising from conference organizers to hear about conferences they wish to attend. This tool would streamline that information transfer and registration process. The targeted users in the system are:

- Regular users (students, professors...etc). These are our main users. The user will be able to browse different subjects and all the conferences related to the selected subject. Then the user will have the choice to register through our system which will work in this case as the third party between the conference and the user.

2 Assumptions

Indeed, we built our project on several assumptions. All of which are:

- In reality, most conferences offer different ticket options such as V.I.P. All Access pass or discounted student passes. These different options attempt to attract potential conference attendees from various backgrounds. For the purposes of this project, we assume that each conference only offers one standard ticket.
- Each user can register for one conference at a time.
- Users must create an account to register for a conference.
- No user can add or edit a contribution. The website shows a predefined list of contributions. This is based on the assumption that the list is given to us by the conference organizers.
- An event can only occur at one conference.
- A contribution is only associated with one conference.
- All the events are in the same location as the conference.
- Maximum capacity for each event at a conference is not capped to a number smaller than the maximum capacity for the conference. In other words, any event does not have limited seats. This is different from conferences in real life where some events have a limited capacity and is first come first served.

3 Graphical Schema for the Database and Its Relations

As you can see from the (fig.1) below, we have nine entities (tables) some of which are associative tables and the others are main entities.

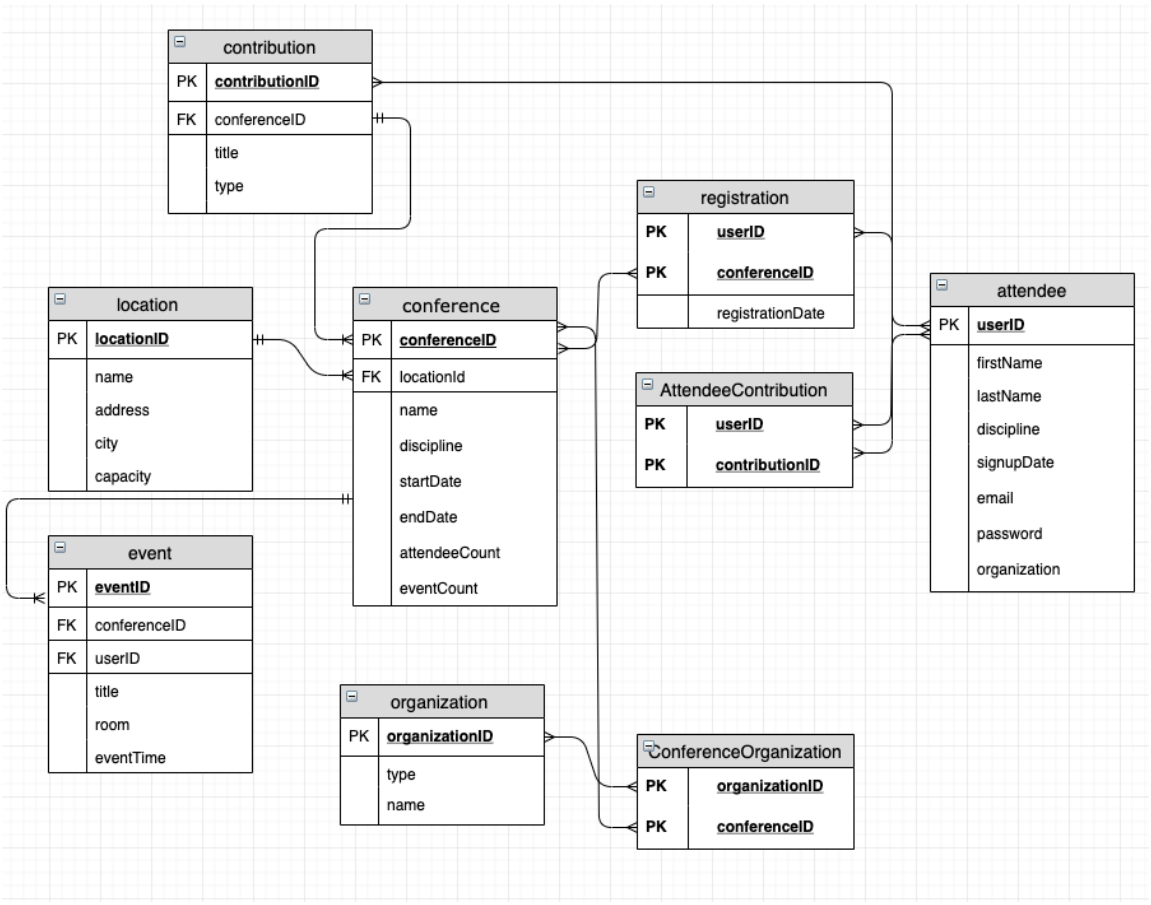

Figure 1: Entity Relationship Diagram.

4 Data Definition Language Statements

As described in the previous section, we have ten relations, most of which stand as a table, and other as an associative "junction" table. In our work, we tried to achieve the second normal form (2NF) as much as we can. The following DDL shows how much we achieved in this manner.

- Create The Database:

```
CREATE DATABASE IF NOT EXISTS `ConferenceGate`
DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
USE `ConferenceGate`;
```

- Create The Main Entities:

```
CREATE TABLE IF NOT EXISTS `Attendee` (
  `userID` int(11) NOT NULL,
  `firstName` varchar(250) NOT NULL,
  `lastName` varchar(250) NOT NULL,
  `discipline` varchar(2500) DEFAULT NULL,
  `organization` varchar(250) NOT NULL,
  `signUpDate` datetime NOT NULL,
  `email` varchar(250) NOT NULL,
  `password` varchar(100) DEFAULT NULL,
  `username` varchar(50) DEFAULT NULL
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `Conference` (
  `conferenceID` int(11) NOT NULL,
  `name` varchar(25000) NOT NULL,
  `locationID` int(11) DEFAULT NULL,
  `discipline` varchar(2500) DEFAULT NULL,
  `startDate` date DEFAULT NULL,
  `endDate` date DEFAULT NULL,
  `attendeeCount` int(11) DEFAULT NULL,
  `eventCount` int(11) DEFAULT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2708 DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `Contribution` (
  `contributionID` int(11) NOT NULL,
  `title` varchar(2500) NOT NULL,
  `type` varchar(250) NOT NULL,
  `conferenceID` int(11) DEFAULT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
```

```
CREATE TABLE IF NOT EXISTS ‘Event‘ (
  ‘eventID‘ int(11) NOT NULL,
  ‘title‘ varchar(2500) NOT NULL,
  ‘conferenceID‘ int(11) NOT NULL,
  ‘userID‘ int(11) DEFAULT NULL,
  ‘room‘ varchar(10) NOT NULL,
  ‘eventTime‘ datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS ‘Location‘ (
  ‘locationID‘ int(11) NOT NULL,
  ‘name‘ varchar(2500) NOT NULL,
  ‘address‘ varchar(2500) NOT NULL,
  ‘city‘ varchar(250) NOT NULL,
  ‘capacity‘ int(11) DEFAULT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1410 DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS ‘Organization‘ (
  ‘organizationID‘ int(11) NOT NULL,
  ‘name‘ varchar(2500) NOT NULL,
  ‘type‘ varchar(2500) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
```

- Create The Junction Entities:

```
CREATE TABLE IF NOT EXISTS ‘AttendeeContribution‘ (
  ‘userID‘ int(11) NOT NULL DEFAULT ’0’,
  ‘contributionID‘ int(11) NOT NULL DEFAULT ’0’
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


CREATE TABLE IF NOT EXISTS ‘ConferenceOrganization‘ (
  ‘conferenceID‘ int(11) NOT NULL DEFAULT ’0’,
  ‘organizationID‘ int(11) NOT NULL DEFAULT ’0’
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


DROP TABLE IF EXISTS ‘ConferenceOrganization‘;
CREATE TABLE IF NOT EXISTS ‘ConferenceOrganization‘ (
  ‘conferenceID‘ int(11) NOT NULL DEFAULT ’0’,
  ‘organizationID‘ int(11) NOT NULL DEFAULT ’0’
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS ‘Registration‘ (
  ‘userID‘ int(11) NOT NULL DEFAULT ’0’,
  ‘conferenceID‘ int(11) NOT NULL DEFAULT ’0’,
  ‘registrationDate‘ datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- Keys and Indices

```
ALTER TABLE ‘Attendee‘
  ADD PRIMARY KEY (‘userID‘);

ALTER TABLE ‘AttendeeContribution‘
  ADD PRIMARY KEY (‘userID‘,‘contributionID‘);

ALTER TABLE ‘Conference‘
  ADD PRIMARY KEY (‘conferenceID‘),
  ADD KEY ‘locationID‘ (‘locationID‘);

ALTER TABLE ‘ConferenceContribution‘
  ADD PRIMARY KEY (‘conferenceID‘,‘contributionID‘);

ALTER TABLE ‘ConferenceOrganization‘
  ADD PRIMARY KEY (‘conferenceID‘,‘organizationID‘);
```

```
ALTER TABLE `Attendee`
  ADD PRIMARY KEY (`userID`);

ALTER TABLE `AttendeeContribution`
  ADD PRIMARY KEY (`userID`,`contributionID`);

ALTER TABLE `Conference`
  ADD PRIMARY KEY (`conferenceID`),
  ADD KEY `locationID` (`locationID`);

ALTER TABLE `ConferenceOrganization`
  ADD PRIMARY KEY (`conferenceID`,`organizationID`);

ALTER TABLE `Contribution`
  ADD PRIMARY KEY (`contributionID`),
  ADD KEY `conferenceID` (`conferenceID`);

ALTER TABLE `Event`
  ADD PRIMARY KEY (`eventID`),
  ADD KEY `conferenceID` (`conferenceID`);

ALTER TABLE `Location`
  ADD PRIMARY KEY (`locationID`);

ALTER TABLE `Organization`
  ADD PRIMARY KEY (`organizationID`);

ALTER TABLE `Registration`
  ADD PRIMARY KEY (`userID`,`conferenceID`),
  ADD KEY `conferenceID_FK` (`conferenceID`);

ALTER TABLE `Attendee`
  MODIFY `userID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=7;

ALTER TABLE `Conference`
  MODIFY `conferenceID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2708;

ALTER TABLE `Contribution`
  MODIFY `contributionID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;

ALTER TABLE `Event`
  MODIFY `eventID` int(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `Location`
  MODIFY `locationID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=1410;
```

```
ALTER TABLE `Organization`
  MODIFY `organizationID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;

ALTER TABLE `Conference`
  ADD CONSTRAINT `conference_ibfk_1` FOREIGN KEY (`locationID`)
  REFERENCES `Location` (`locationID`) ON DELETE CASCADE;

ALTER TABLE `Contribution`
  ADD CONSTRAINT `contribution_ibfk_1` FOREIGN KEY (`conferenceID`)
  REFERENCES `Conference` (`conferenceID`);

ALTER TABLE `Event`
  ADD CONSTRAINT `event_ibfk_1` FOREIGN KEY (`conferenceID`)
  REFERENCES `Conference` (`conferenceID`) ON DELETE CASCADE;

ALTER TABLE `Registration`
  ADD CONSTRAINT `conferenceID_FK` FOREIGN KEY (`conferenceID`)
  REFERENCES `Conference` (`conferenceID`),
  ADD CONSTRAINT `userID_FK` FOREIGN KEY (`userID`) REFERENCES `Attendee` (`userID`);
```

5 Front-End and Back-End Design

We used several front-end and back-end tools to build our system. All the tools that have been used are displayed in (fig.2 and fig.3).

Figure 2: Front-End Tools.

Figure 3: Back-End Tools.

6 Implementation

In this section, we are going to walk through the system by showing the pseudo-code along with screenshots from the system.

1. The main page of our system, which contains the main functions and links, which can help the user to navigate through the our system, besides some aggregation functions.

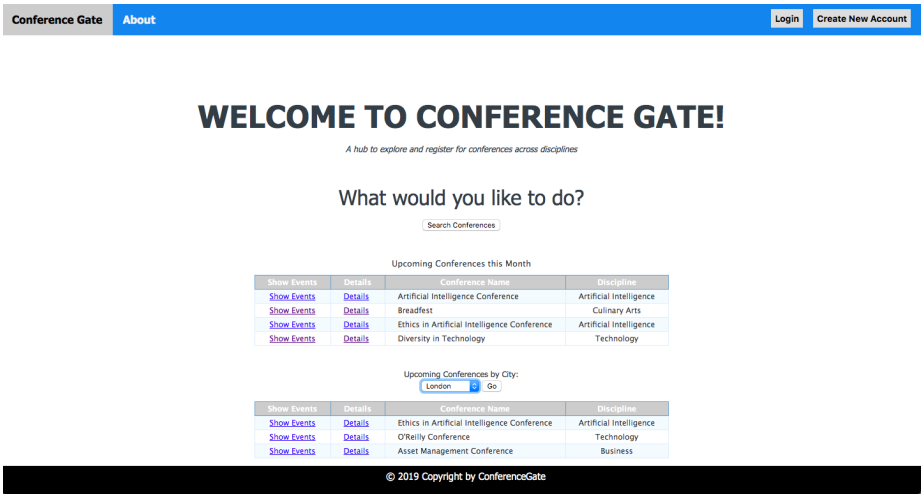

Figure 4: Main Page.

2. One of the main functions is help the user create an account. So, we create a function that can help the user to do so.

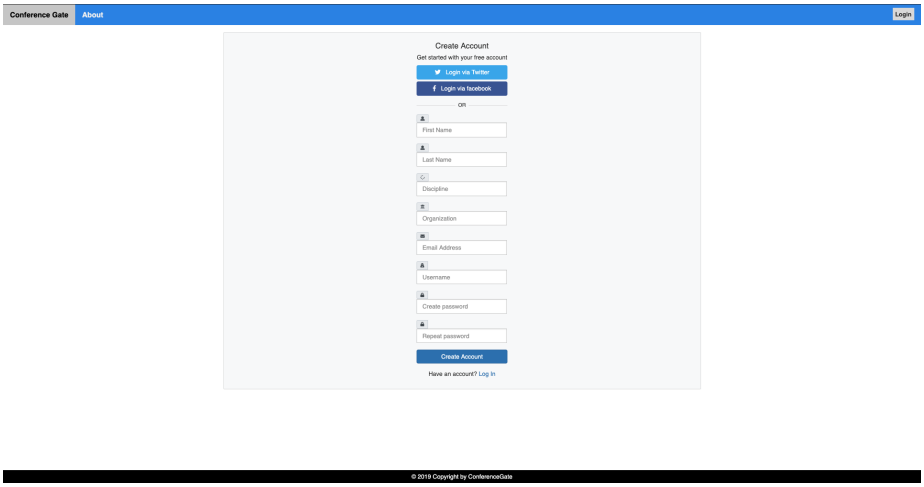

Figure 5: Create an Account.

3. If a user already has an account, the user can basically just login.

Figure 6: Login.

4. Our system supports different kinds of search function, which can help the user to further investigate. Here is an example for searching for conferences located in the city of 'Halifax', followed by the result.

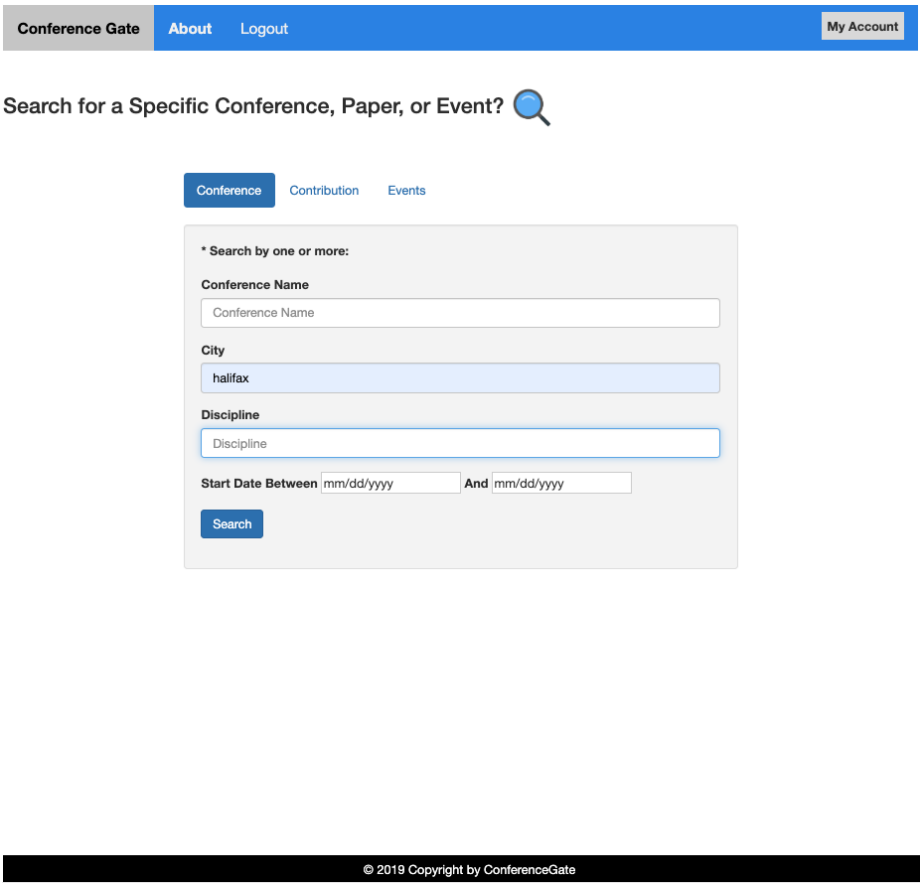

Figure 7: Conference Search.

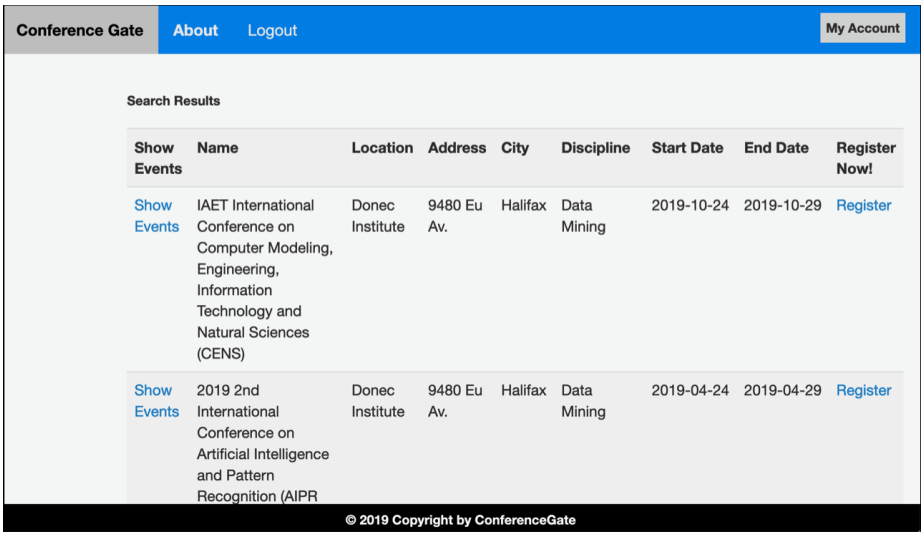

Figure 8: Conference Search Results.

Additionally, there are more ways to search, like by specific paper or event (fig. 9 and fig. 10).

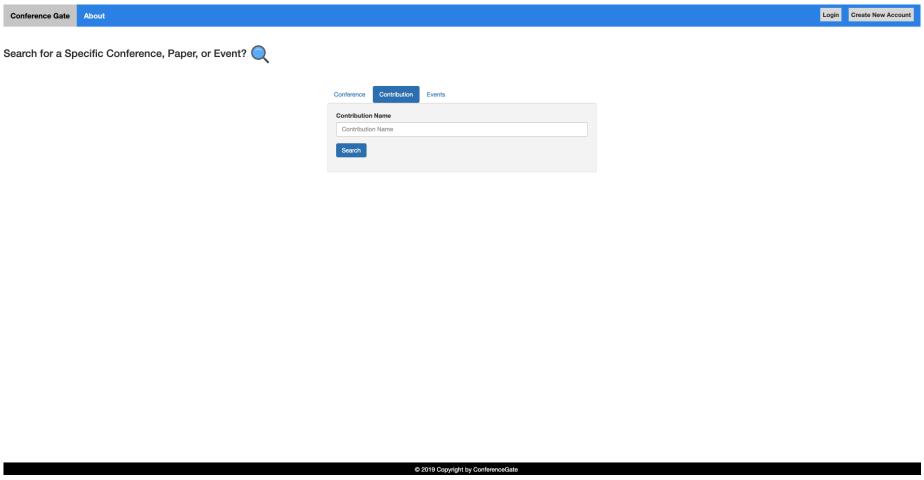

Figure 9: Contribution Search.

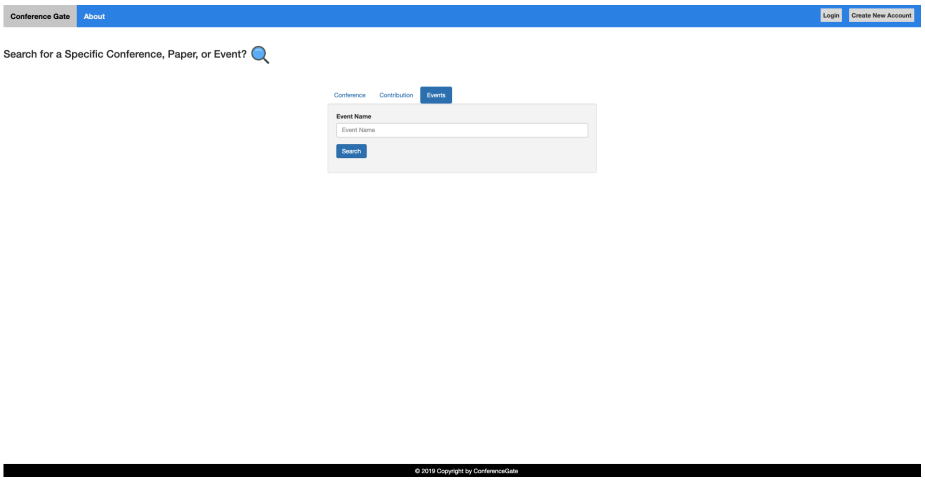

Figure 10: Event Search.

5. Our system supports a user-friendly, elegant payment gateway so that the user can register for the conference through this platform instead of going to the official conference page.

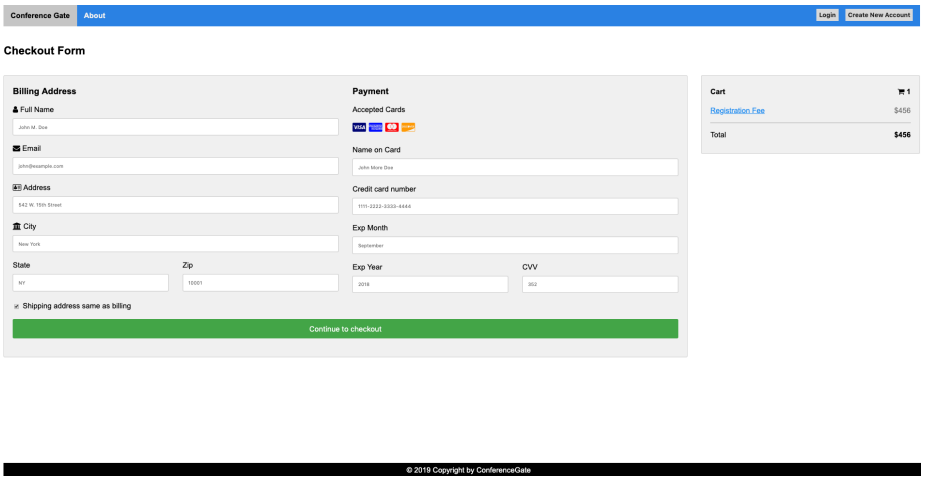

Figure 11: Registration Form.

Here, our system confirmed the registration since the process passes all the conditions.

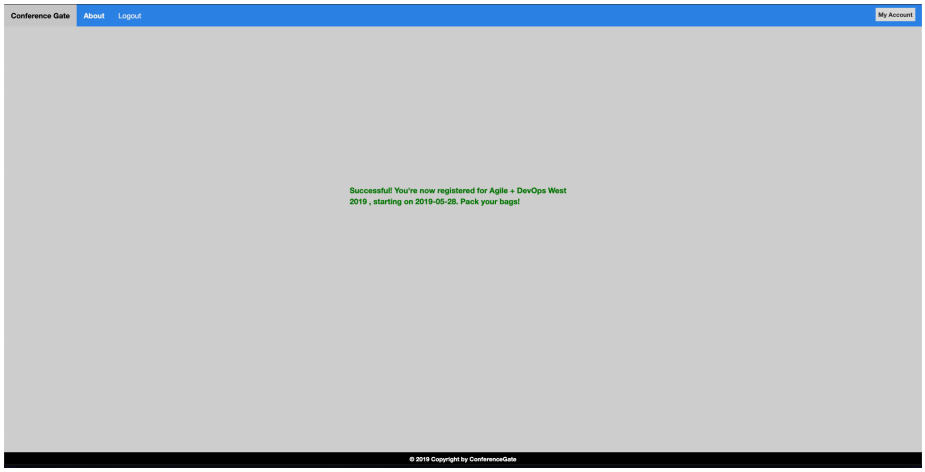

Figure 12: Successful Registration.

6. Moreover, our system supports a smart user page (dashboard) which shows the user's personal information and the conferences that he/she registered for.

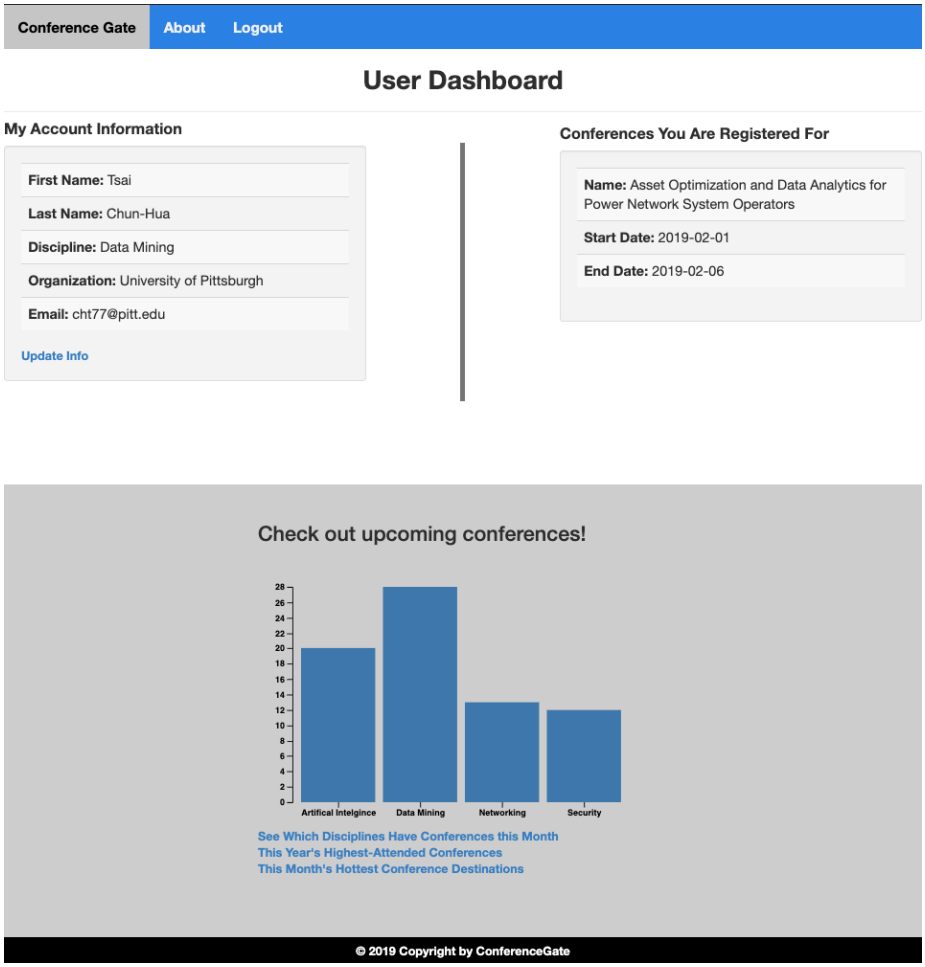

Figure 13: User Page.

6.1 Operations

- **Browsing Function**. In our system we have couple of ways to browse through conferences, one we already mentioned in the previous section, and the other one is (fig.14) below.

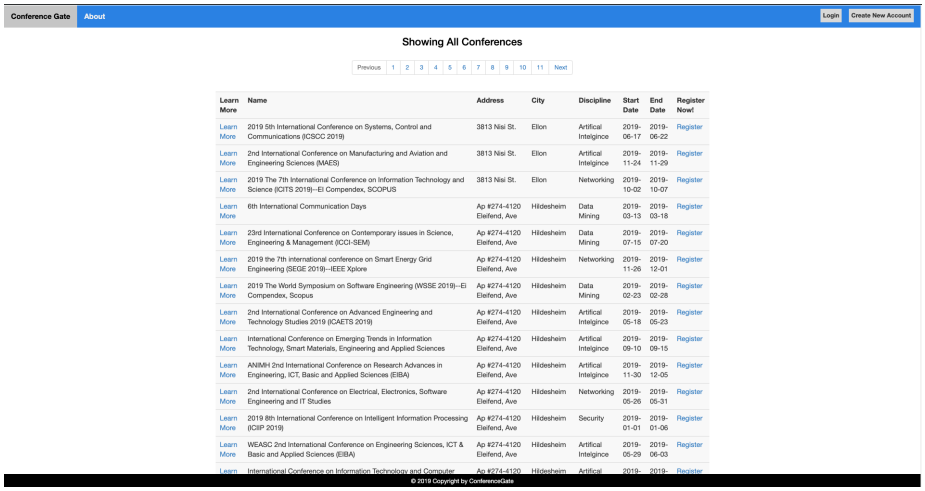

Figure 14: Browsing Conferences.

- **Update Function**. Our users have the ability to update their information through a link in their account page (fig. 13), which will redirect them to the update information page (fig. 15)

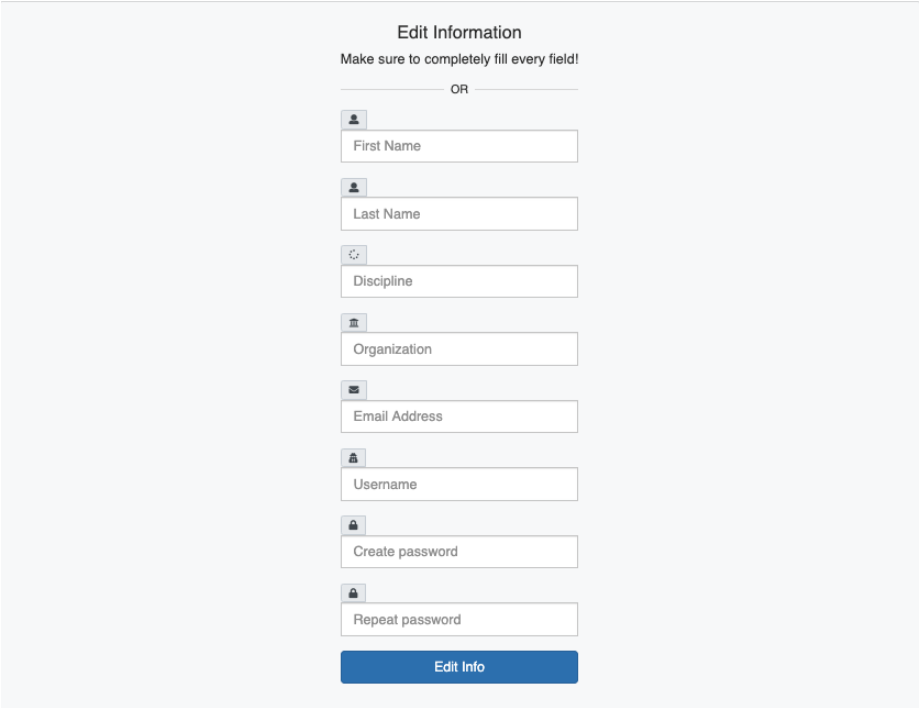

Figure 15: Update User Information Page.

- **Error Checking, Security, and Testing Functions**.
  In order to have more robust system, we identify several conditions and criteria that can help us to control our system. For example, the user will not be able to login if there is a problem with either the username or password. Also, if the conference is full, user will not able to register and a error message will appear (fig. 17). In addition to that, if the user already registered for a conference, the system will not let him/her register again (fig. 18). One of main security procedures that we have considered is that the password has to be encrypted using hash-encryption function (fig. 19).

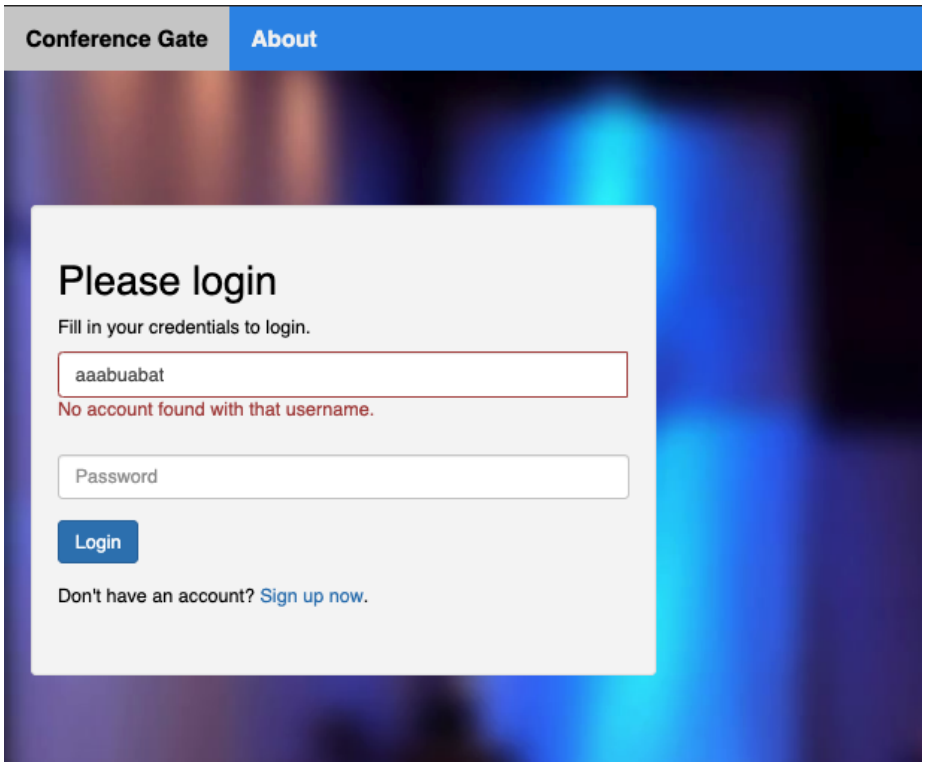

Figure 16: Login Error Message.

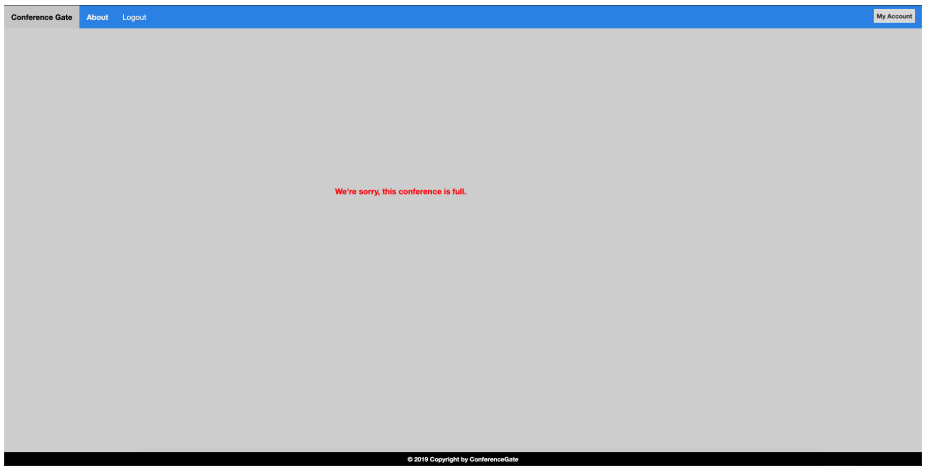

Figure 17: Conference is Full Error Message.

Figure 18: "Already Registered" Error Message.

Figure 19: How the Password looks like in Our Database.

- **Data Aggregation**
  - **First Aggregation Function**. This function is located in the landing page, which can help any guest to browse through the upcoming events and conferences.

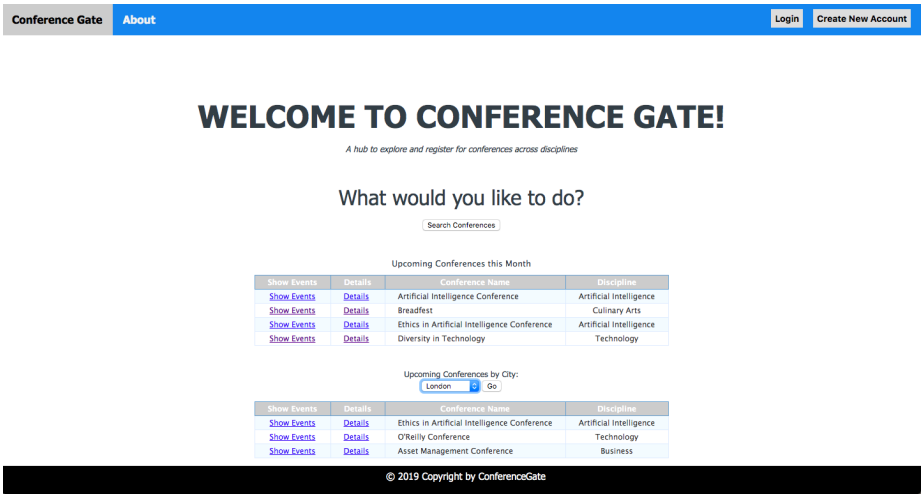

Figure 20: First Aggregation Function.

– **Second Aggregation Function**. This function is located in the user page, which can help the user to see the hottest conference destinations.

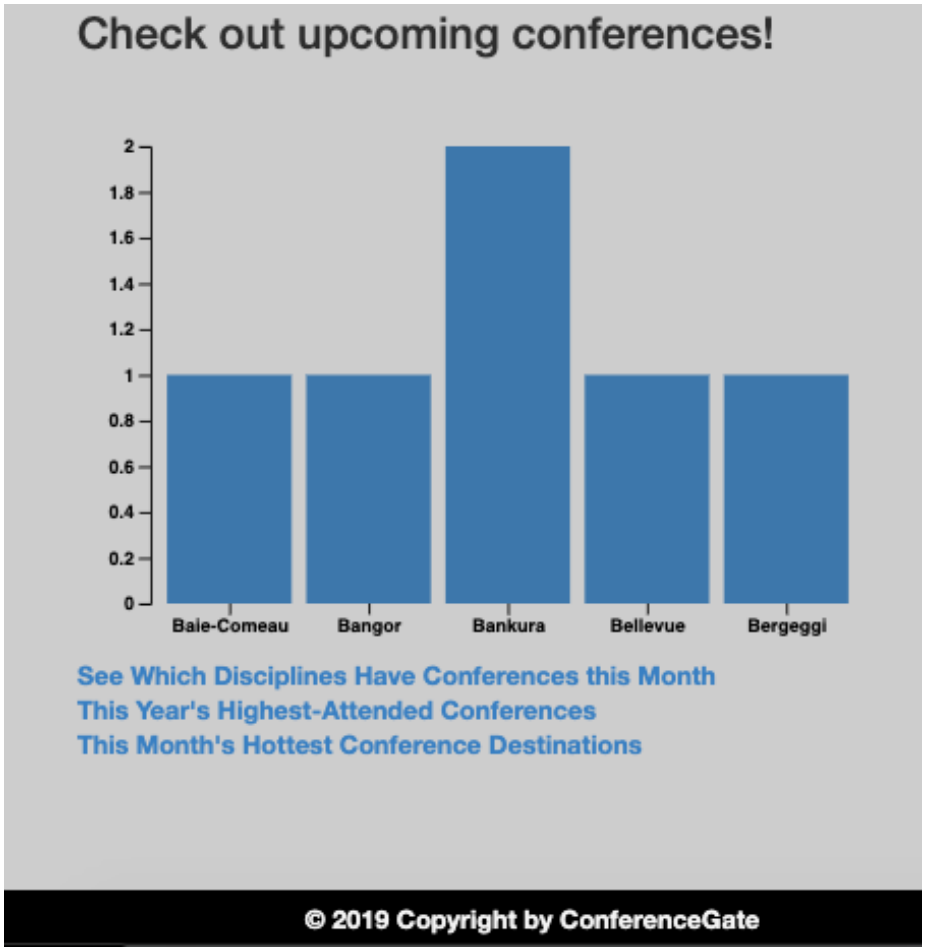

Figure 21: Hottest Conference Destinations.

– **Third Aggregation Function**. This function is located in the user page, which can help the user to see disciplines that conferences have this month.

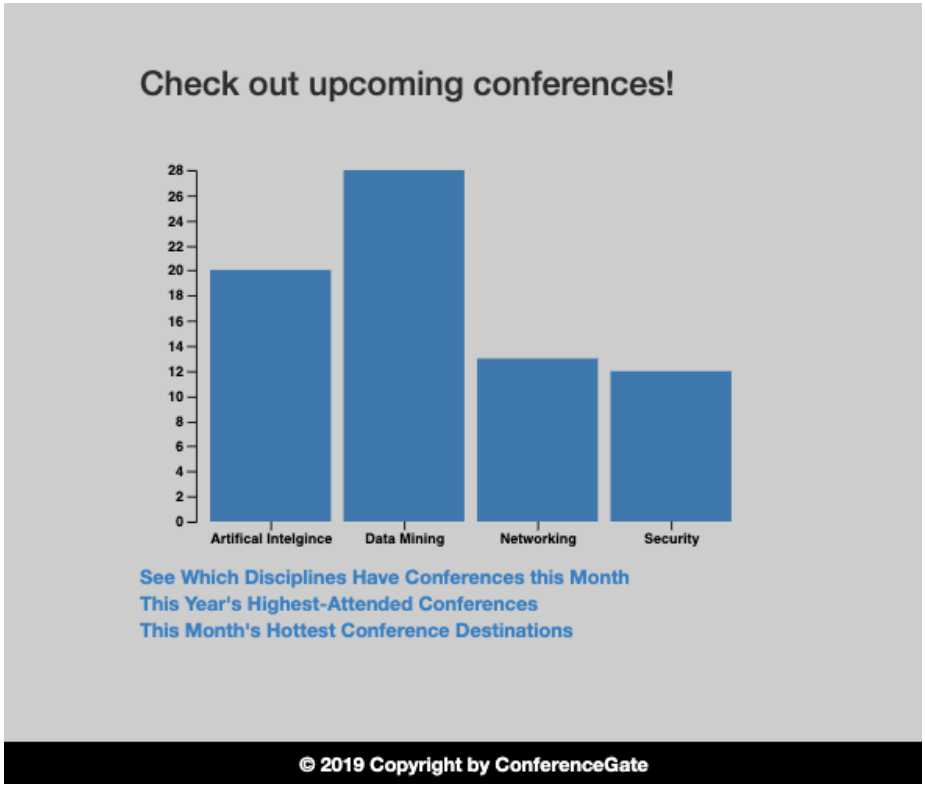

Figure 22: Disciplines that the Conferences Have this Month.

7 Improvements and Conclusion

To conclude, our system allows users to browse through and register for conferences across disciplines. Although our system has several important and useful features, there is room for further improvements. Our future work will be focused on how to have different kinds of users in which each kind has different features and functions, such as organizations representatives and administration users. Also, we seek to have more intelligent system that will recommend conferences to users based on various factors such as past browsing history. Furthermore, other features such as that the user can 'like' and 'save' conferences he/she is interested in as well as feature that will give the user an alert before the registration due date are some ideas for potential improvements.

— The End —