

# Duke Conversations

CS 316

NOAH BURRELL, ANNE DRISCOLL, KIMBERLY EDDLEMAN, SUMMER SMITH,  
SARP UNER

## Application

### The Problem

Duke Conversations is a program run through the provost's office, not through the UCAE, that holds dinners twice or three times a dinner with faculty members, as a way to bring together the Duke community. The goal of the program is to create more small group interaction between faculty members and students, and create an atmosphere of academic engagement.

Currently, the program accepts dinner applications through google forms, where individuals on the planning committee accept or reject applicants. The intent is to accept people who haven't been to the program before, and create a dinner that reflects Duke's diversity. Unfortunately, because different members of the planning committee work on different dinners, and may not be familiar with the names from previous dinners that they didn't work on, people can slip through the cracks. That can take the form of people who may have applied many times and never been accepted, people who have consistently been accepted, or people who have skipped previous dinners and continue to be accepted.

### The Application

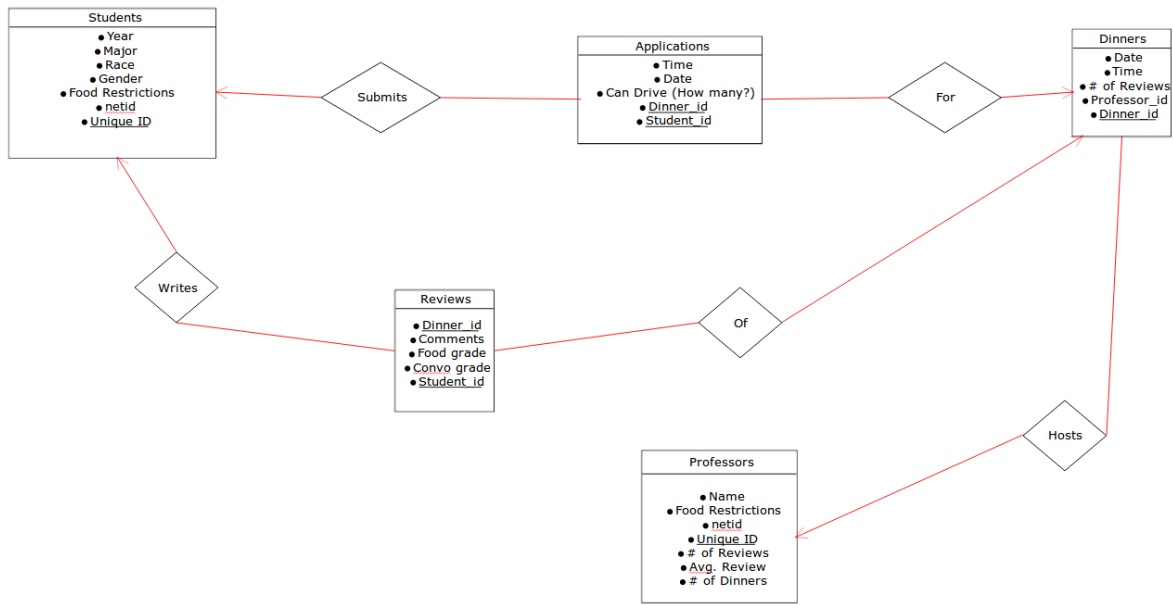
The application we intend to build will help the organizing team keep track of attendees. By keeping count of how many times each person has applied, and if they've attended previous dinners, we'll be able to ensure that the selection process is standardized, and the team won't have to try to remember the names of all the people that went to (or applied to) dinners that they didn't manage.

To do that, we'll have to create a login process for both sets of parties, a way for people to apply, and a way for planners to select applicants.

## Database Design

The ER diagram for the database is provided below. Similar to the convention in the book, the diagram shows many-one relationships with referential integrity constraints with an open arrow (in place of a curved arrow, as a curved arrow was not available in the tool used to draw these diagrams).

Note that the referential integrity constraints are necessary, as, for example, a non-existent student in the database should not be able to submit an application for a non-existent dinner. Also note, however, that our database does not assume that a student necessarily has to have submitted an application, which is reasonable. Similarly, it does not expect a professor to host a dinner to exist in the database, which would be limiting, as a user may wish to register first and then host/apply for a dinner later.



Following are the relations derived from this ER diagram. Relations for many-one relationships were combined with the appropriate entity sets as suggested. For instance, the relation Applications combines the relations for the relationships Submits and For. Applications could have also been represented as a weak entity set in the ER diagram above, but this version of the ER diagram lends itself to an easier interpretation.

Also note that even though all the attributes above the line in each relation denote keys, they are all shown as primary keys (PK). However, in those cases where there is more than one key, the combination of all the attributes in the key section should be interpreted as constituting a key, as opposed to each attribute being a key itself. To give an example, consider the relation Reviews. Reviews shows both Dinner\_id and Student\_id as keys, though neither constitutes a key by itself, since a student can submit applications for multiple dinners and a dinner may have applications from multiple students. Therefore, the key for the relation Applications is (Dinner\_id, Student\_id).

Note that, for the Dinners relation below, Dinner\_id is a unique identifier of a specific dinner (a key), possibly a sequential id for dinners. However, depending on later implementation specific decisions, "Date, Time, Professor\_id" can also be used as a key for Dinners, as we would not expect a professor to host two different dinners at the same exact day and time.

| Students           |          |    | Professors        |          |      |
|--------------------|----------|----|-------------------|----------|------|
| Unique ID          | int      | PK | Unique ID         | int      | PK   |
| Name               | varchar  |    | Name              | varchar  |      |
| Food Restrictions  | varchar  |    | Food Restrictions | varchar  |      |
| netid              | varchar  |    | netid             | varchar  |      |
| Applied            | int      |    | # of Reviews      | smallint |      |
| Accepted           | int      |    | Avg. Review       | smallint |      |
| Year               | int      |    | # of Dinners      | smallint |      |
| Major              | varchar  |    |                   |          |      |
| Gender             | varchar  |    |                   |          |      |
| Dinners            |          |    | Applications      |          |      |
| Dinner_id          |          | PK | Student_id        | int      | PK   |
| Date               | date     |    | Dinner_id         | int      | PK   |
| Time               | time     |    | Drive_num         | int      |      |
| # of Reviews       | int      |    | Selected          | boolean  | NULL |
| Professor_id       | int      |    | Time              | time     |      |
|                    |          |    | Date              | date     |      |
| Reviews            |          |    |                   |          |      |
| Dinner_id          | int      | PK |                   |          |      |
| Student_id         | int      | PK |                   |          |      |
| Comments           | varchar  |    |                   |          |      |
| Food Grade         | smallint |    |                   |          |      |
| Conversation Grade | smallint |    |                   |          |      |

Now, let us check to see if the relations are in BCNF. For Students, the only FD that exists is “Unique ID → all other attributes”. It is self evident that the left side of this FD is a super-key, and thus Students is in BCNF: no further decomposition is necessary/possible.

Next, for Professors, similarly, the only FD is “Unique ID → all other attributes”, which, again, is in BCNF since the left hand side of this FD is a super-key.

For Dinners, there are two FDs. First one, similar to the previous ones, is “Dinner\_id → all other attributes”. This FD does not violate BCNF. The next one is “Date, Time, Professor\_id → Dinner\_id, # of Reviews”, as noted above. This FD also does not violate BCNF, as the left hand side is a super-key.

Similar arguments show that FDs for Applications and Reviews also do not violate BCNF, and thus our relations are in BCNF.

## Sample Data

### Past Data

Until now, all the data that Duke Conversations has collected has been in the form of applications to dinners by individuals. These applications are sent through Google Forms, with a separate form for each dinner, with mostly the same questions. The forms being separate, with no way to aggregate the data for each individual is what causes the inconsistencies in application acceptance.

Below see a sample of applications for a given dinner:

| Timestamp          | Name     | Netid | Phone       | Year | Diet   |
|--------------------|----------|-------|-------------|------|--------|
| 8/31/2016 21:57:05 | Michelle | m123  | 000-000-000 | 2018 | none   |
| 8/31/2016 21:58:19 | Ibrahim  | i123  | 000-000-000 | 2020 | Veggie |

Over times the questions asked have varied, including gender, major, and ability to drive to the event. For each event they apply to, applicants include the data for those variables, leading to lots of inconsistencies in the data over time.

### Current Data

We have compiled (for the preliminary report) the applications from a few of the dinners last semester, totaling 600 applications. This sample set includes variables for all the past questions, with NA's for the applications that were not asked that question originally.

In this sample set, we have data that includes many repeat attendees, and many inconsistencies that need to be corrected to be added to the database.

## Data Assumptions

As seen in our check statements<sup>1</sup> there are several conditions that must be held up in the data, the way we have structured it. Much of the data references data held in other tables, and those relationships must be upheld. As examples, we walk through several of our testing statements.

- The first insert statement checks that the student\_ID associated with a review belongs to a student who actually applied for and attended the dinner that he is reviewing. For this reason, it has a student\_id that does not exist in the Students table.
- The next insert statement updates a key in the applications table.
- We test that it's not possible to reference a professor that doesn't exist in the dinners table.
- We check that it's not possible to review a dinner that never happened.
- There are also insert statements that create a review written by a student that does not exist in the database, assign a nonexistent major to a student, and create a student entry with an invalid netID.

## Final Interface Goals

### Users

The user interface caters to two types of users: attendees, who use the site to sign up for events and give reviews of the events, and admins, who manage the events that can be signed up for, look at user/event statistics, and generate the list of people who can attend.

As such, the user interface design has two main views, a public view, accessible to anyone, and an admin view, accessible with a password or other login system.

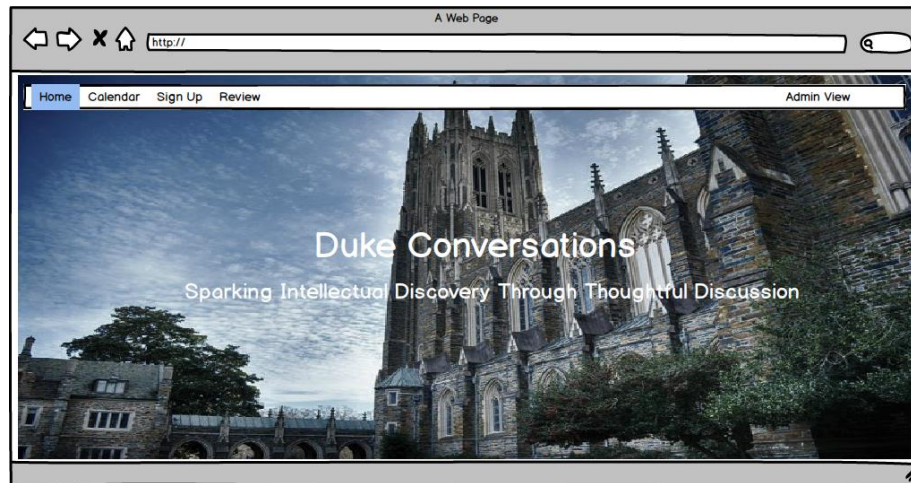
---

<sup>1</sup> See InsertCheck.sql

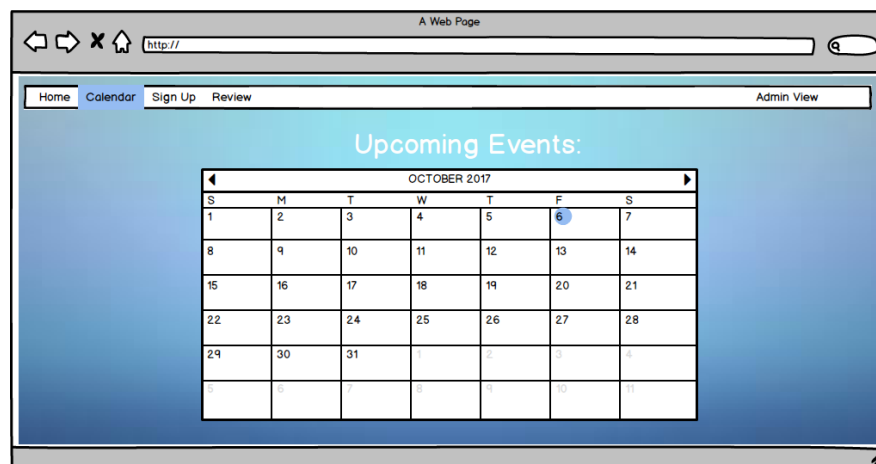
## Design

The website is designed as a multi-page site with tabs at the top of every page to navigate around.

The public view has 4 main tabs, Home, Calendar, Sign Up and Review. It also has an “Admin View” button which can be used to switch over to the admin view.



The calendar, which may or may not be implemented, depending on the final needs of the group, will include a visual view of all the upcoming event.



The sign up page will be where attendees can choose which event they want to sign up for. In implantation, the design will be an actual form with more information than shown below.

A Web Page

http://

Home Calendar Sign Up Review Admin View

### Sign Up for a Dinner:

Event: Event 1, Event 2, Event 3

Name: [text input]

Email: [text input]

Year: [text input]

Phone number: [text input]

Any Special Dietary Needs?: [text input]

Can you drive?: ☒ Yes, ☐ No, ☐ Possibly

If so how many?: [text input]

Sign Up

The sign up confirmation page will let users know that they have submitted their forms.

A Web Page

http://

Home Calendar Sign Up Review Admin View

### Sign Up for a Dinner:

Your Sign Up for **EVENT** has been submitted!

If you are invited to attend you will receive an email shortly

The review tab will be similar to the sign up tab, but with questions about how their experience at the dinner was, rather than an application for the dinner.

A Web Page

http://

Home Calendar Sign Up Review Admin View

### Review a Dinner:

Event: Event 1, Event 2, Event 3

Name: [text input]

Quality of Dinner: [slider]

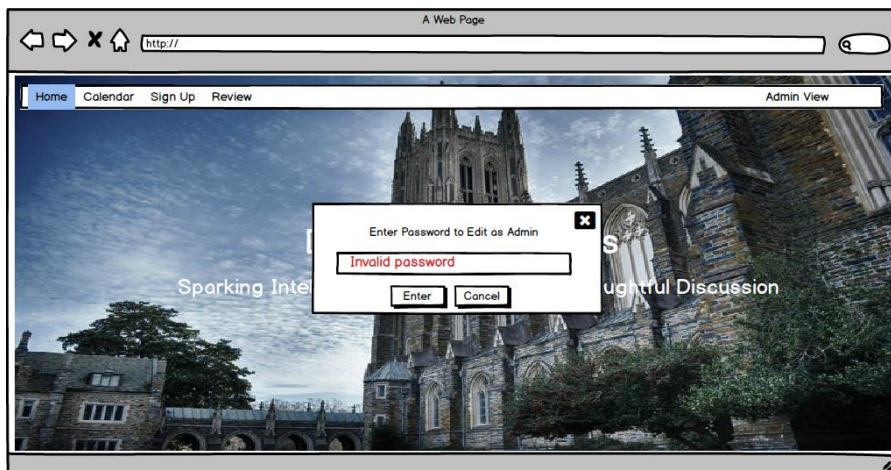
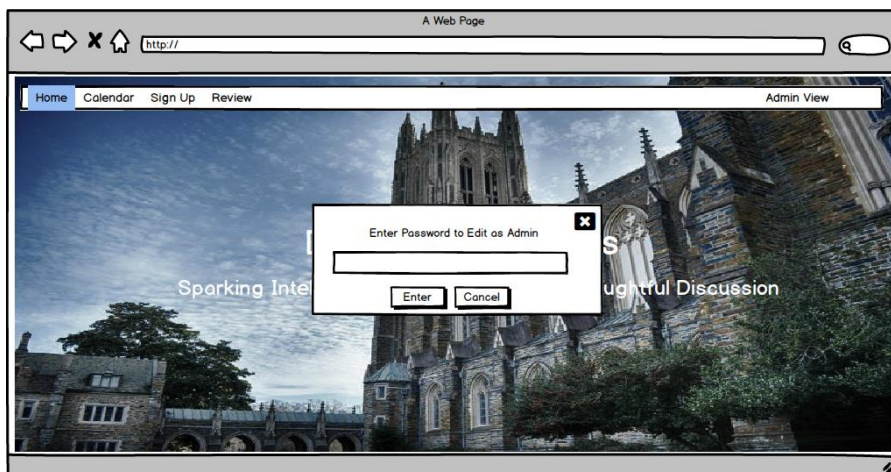
Comments: [text area]

Submit

The review confirmation page will confirm the actual submitted review.

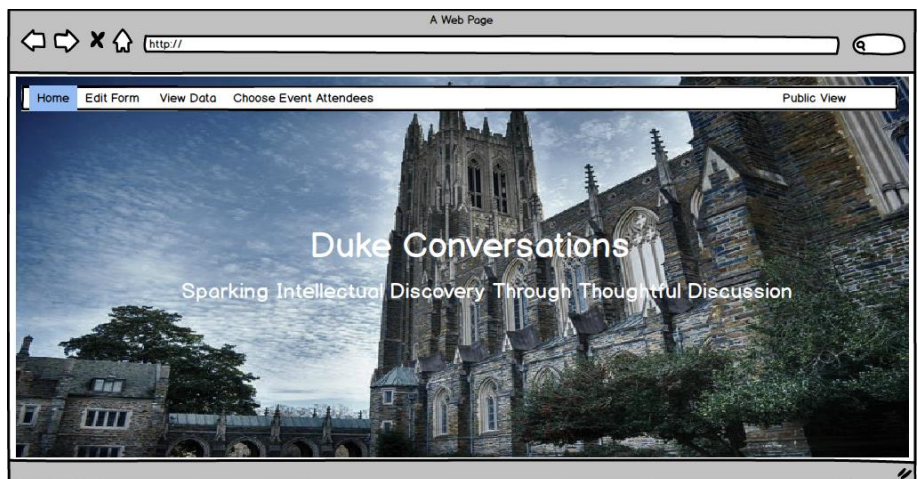


On the admin switch page, the user will be prompted with a box asking for a password. If the correct password is entered, the view will transition to the homepage of the Admin view. Otherwise, the box will tell the user they entered the correct password. In implementation, a more advanced login might be used to keep data more secure.

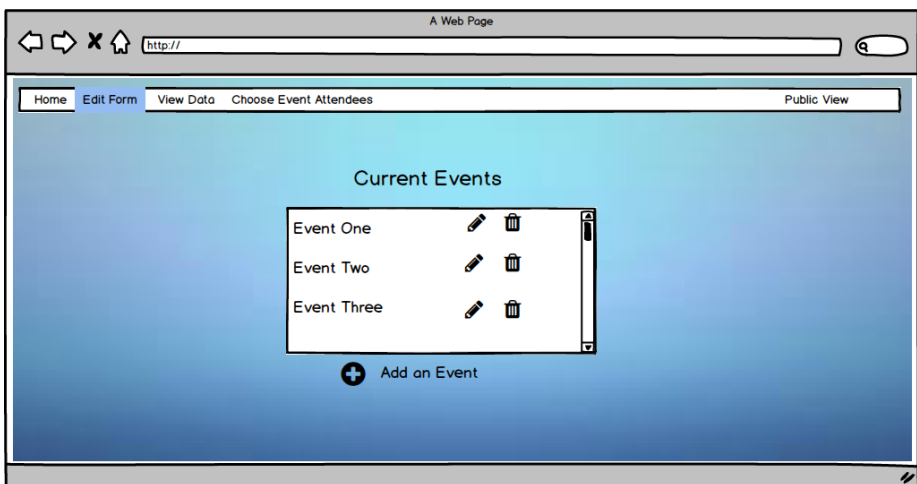




The Admin View is what allows administrators to view the data, change the events it is possible to sign up for, and generate a list of people to attend the events based on various criteria. It has 4 main tabs, Home, Edit Form, View Data, and Choose Event Attendees. It also has a button to return to the public view.



The edit form tab allows for the form to be updated to include new events to be signed up for or remove events that no longer can be signed up for. Each event currently in the sign up can be edited by clicking on the pencil or deleted by clicking on the trashcan. New events can be added with the plus.




A Web Page

http://

Home Edit Form View Data Choose Event Attendees Public View

### Edit Event

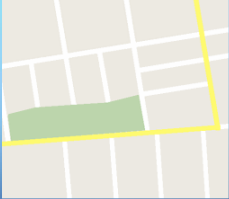
Event Name

Date  

Time

Location

Description




A Web Page

http://

Home Edit Form View Data Choose Event Attendees Public View

### Add Event


Event Name

Date  

Time

Location

Description



The View Data tab allows admins to see all the data organized in a neat form. The data is organized into four categories: Students (all the students who have filled out at least one of the sign ups), Professors, Events (each dinner that has or will be happening), and Applications (a list of each application/review that has been submitted). Each entry can be opened up and viewed in more detail (for example, student, professor, event, etc). Additionally each tab can be sorted or filtered by various criteria)

A Web Page

http://

Home Edit Form View Data Choose Event Attendees Public View

Students Professors Events Applications

| Name               | Age | Gender | Major      |
|--------------------|-----|--------|------------|
| Giacomo Guilizzoni | 40  | M      | Music      |
| Marco Botton       | 38  | M      | ECE        |
| Mariah MacLachlan  | 41  | F      | Physics    |
| Valerie Liberty    | 20  | F      | Philosophy |
|                    |     |        |            |
|                    |     |        |            |
|                    |     |        |            |
|                    |     |        |            |
|                    |     |        |            |
|                    |     |        |            |

A Web Page

http://

Home Edit Form View Data Choose Event Attendees Public View

Students Professors Events Applications

Giacomo Guilizzoni

Relevant Details about each person could go here. These could include things like age, major, number of dinner attended, number of reviews, etc.

Applications Accepted/Rejected Attendance Rate

The choose attendees tab will use a variety of algorithms to determine a list of people for a new event. Different specifications can be requested and a list can be generated based off of these specifications. Once the list is created, it can be viewed and confirmed or canceled.

A Web Page

http://

Home Edit Form View Data Choose Event Attendees Public View

Generate List of People to Attend an Event

Event Name

Number of People

Priorities

☐ Least number of events attended

☒ Most variety of majors

☒ Best attendance record

Beth Smith

Rachel Stevens

Bob Roberts