**IoASU** (**Innovation on ASU**)

Team Entrepreneurs

Ishwar Bhat Kattur

Anirban Bhattacharya

Dhivya Raja

Priyanka Pakhale

Arizona State University

Prof. Usha Jagannathan

IFT530

05/01/2017

**Introduction**

Student engagement is a vital aspect of academic life and organizations on campus ensure meaningful involvement outside the curriculum. Our project IoASU aims at involving and engaging the students with their organizations, by providing a mobile application for the users as well as the admins, to interact with their organizations.

The project itself had two main objectives:

1. Build a mobile app which must act as a one stop shop for all of ASU’s Organizations as well as for students to engage and interact with an organization;
2. Reduce and try to eliminate usage of paper in the form of flyers and coupons sent out by the organizations, and use the mobile app for the purpose instead.

**Project Requirements / Goal**

The main goal of the project was to build a mobile app with the support of a robust backend, for the users as well as administrators. The focus is on reducing the usage of paper as well as streamlining the process by enhancing user experience. To accomplish this, we used the technologies listed below.

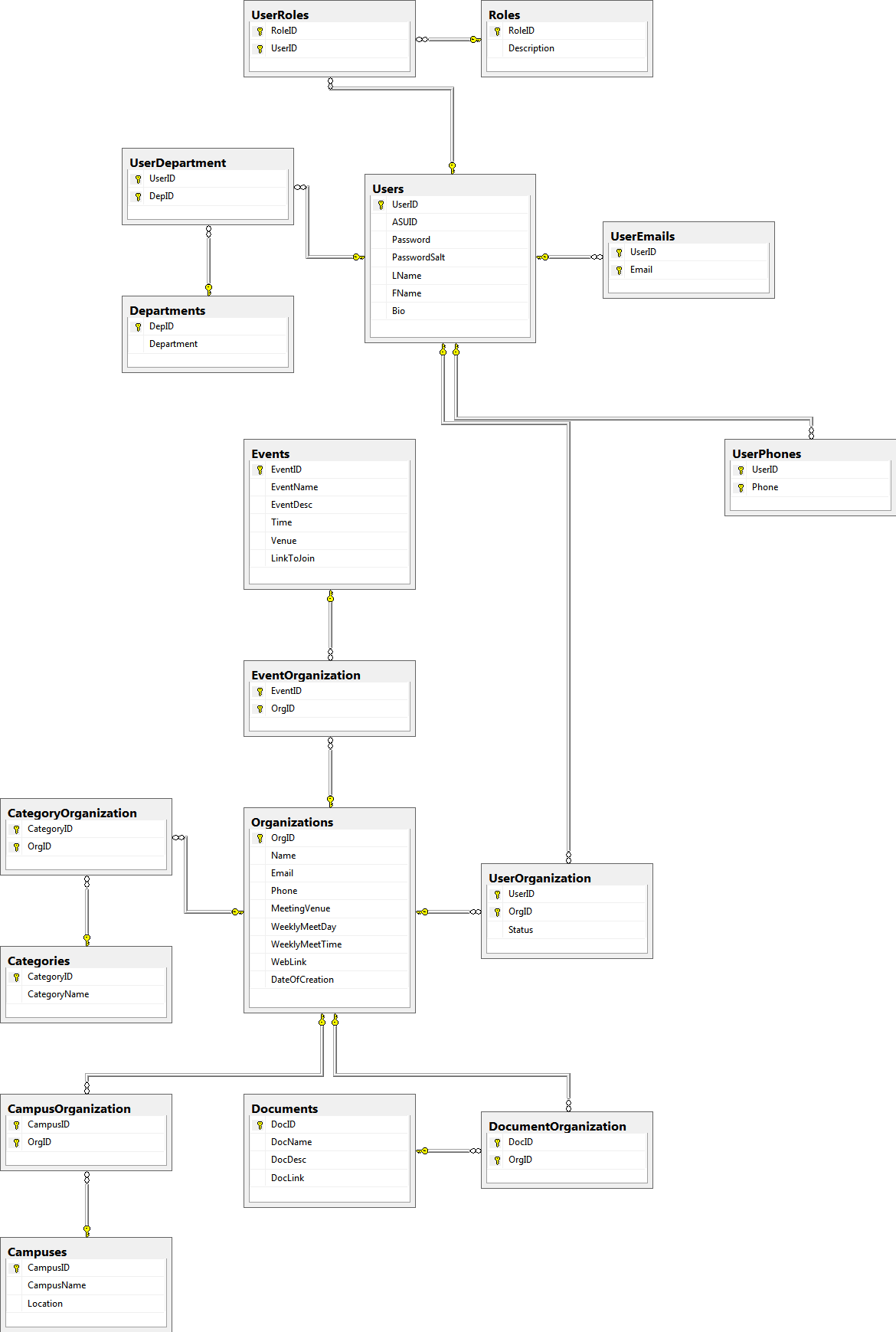
**Technologies used**

Backend: Microsoft SQL Server on cloud (RDS and S3 on Amazon Web Services)

Middleware: Django Framework with Python

Front-end: iOS Mobile Application

Version Control: Git (GitHub) (Publicly available on: https://github.com/IshwarBhat/IoASU)

**Database Design (Database diagram)**

**DDL (Data Definition Language)**

The following are the CREATE DATABASE and CREATE TABLE statements for IoASU. Use of relationship tables with composite keys was a major decision instead of just using foreign keys, as this would definitely lead to more joins. This design feature turned out to be a good judgement call in the project, as the separation of concerns among the tables made querying and understanding the data in general easier.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This script creates the database named IoASU

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

USE master;

GO

IF DB\_ID('IoASU') IS NOT NULL

DROP DATABASE IoASU;

GO

CREATE DATABASE IoASU;

GO

USE IoASU;

-- create the tables for the database

CREATE TABLE Campuses (

CampusID INT PRIMARY KEY IDENTITY,

CampusName VARCHAR(255) NOT NULL,

Location VARCHAR(255) NOT NULL,

);

CREATE TABLE Organizations (

OrgID INT PRIMARY KEY IDENTITY,

Name VARCHAR(255) NOT NULL,

Email VARCHAR(255) NOT NULL,

Phone VARCHAR(10),

MeetingVenue VARCHAR(255),

WeeklyMeetDay VARCHAR(10),

WeeklyMeetTime TIME,

WebLink VARCHAR(255),

DateOfCreation DATETIME DEFAULT GETDATE()

);

CREATE TABLE CampusOrganization (

CampusID INT REFERENCES Campuses (CampusID) NOT NULL,

OrgID INT REFERENCES Organizations (OrgID) NOT NULL,

PRIMARY KEY (CampusID, OrgID)

);

CREATE TABLE Departments (

DepID INT PRIMARY KEY IDENTITY,

Department VARCHAR(255)

);

CREATE TABLE Users (

UserID INT PRIMARY KEY IDENTITY,

ASUID VARCHAR(20),

Password VARBINARY(128) NOT NULL,

PasswordSalt VARCHAR(128),

LName VARCHAR(255) NOT NULL,

FName VARCHAR(255) NOT NULL,

Bio VARCHAR(255)

);

CREATE TABLE UserPhones (

UserID INT REFERENCES Users (UserID) NOT NULL,

Phone VARCHAR(10) NOT NULL,

PRIMARY KEY (UserID, Phone)

);

CREATE TABLE UserEmails (

UserID INT REFERENCES Users (UserID) NOT NULL,

Email VARCHAR(255) NOT NULL,

PRIMARY KEY (UserID, Email)

);

CREATE TABLE UserDepartment (

UserID INT REFERENCES Users (UserID) NOT NULL,

DepID INT REFERENCES Departments (DepID) NOT NULL,

PRIMARY KEY (UserID, DepID)

);

CREATE TABLE UserOrganization (

UserID INT REFERENCES Users (UserID) NOT NULL,

OrgID INT REFERENCES Organizations (OrgID) NOT NULL,

Status VARCHAR(25) NOT NULL,

PRIMARY KEY (UserID, OrgID)

);

CREATE TABLE Events (

EventID INT PRIMARY KEY IDENTITY,

EventName VARCHAR(255) NOT NULL,

EventDesc VARCHAR(255),

Time DATETIME,

Venue VARCHAR(255),

LinkToJoin VARCHAR(255)

);

CREATE TABLE EventOrganization (

EventID INT REFERENCES Events (EventID) NOT NULL,

OrgID INT REFERENCES Organizations (OrgID) NOT NULL,

PRIMARY KEY (EventID, OrgID)

);

CREATE TABLE Categories (

CategoryID INT PRIMARY KEY IDENTITY,

CategoryName VARCHAR(255) NOT NULL,

);

CREATE TABLE CategoryOrganization (

CategoryID INT REFERENCES Categories (CategoryID) NOT NULL,

OrgID INT REFERENCES Organizations (OrgID) NOT NULL,

PRIMARY KEY (CategoryID, OrgID)

);

CREATE TABLE Documents (

DocID INT PRIMARY KEY IDENTITY,

DocName VARCHAR(255),

DocDesc VARCHAR(255),

DocLink VARCHAR(255) NOT NULL

);

CREATE TABLE DocumentOrganization (

DocID INT REFERENCES Documents (DocID) NOT NULL,

OrgID INT REFERENCES Organizations (OrgID) NOT NULL,

PRIMARY KEY (DocID, OrgID)

);

CREATE TABLE Roles (

RoleID INT NOT NULL,

Description VARCHAR(255) NOT NULL,

PRIMARY KEY (RoleID)

);

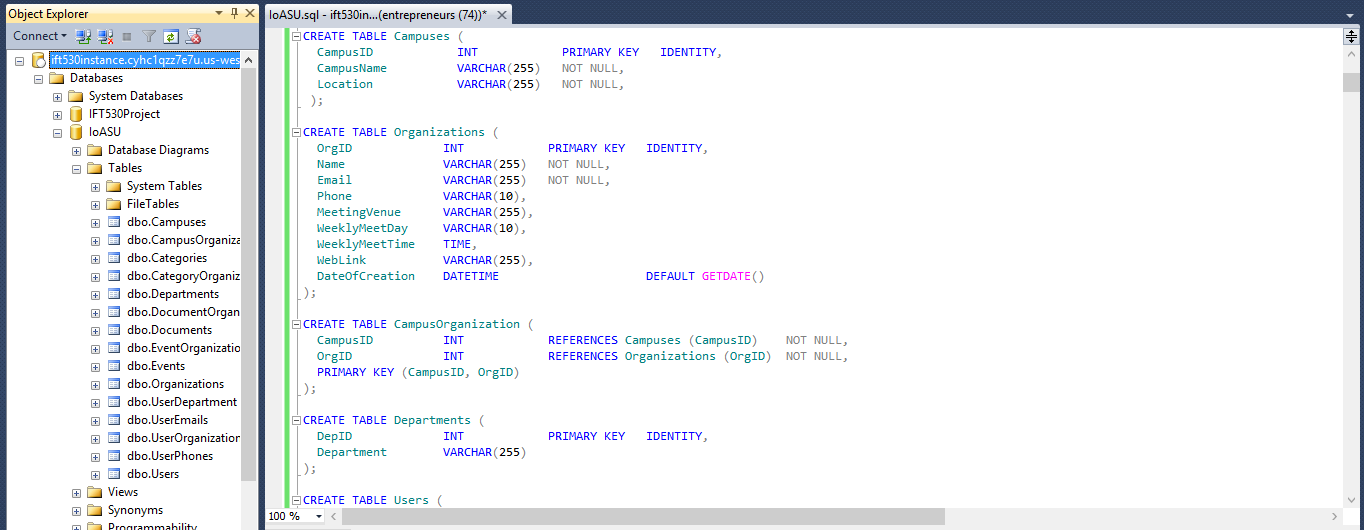
CREATE TABLE UserRoles (

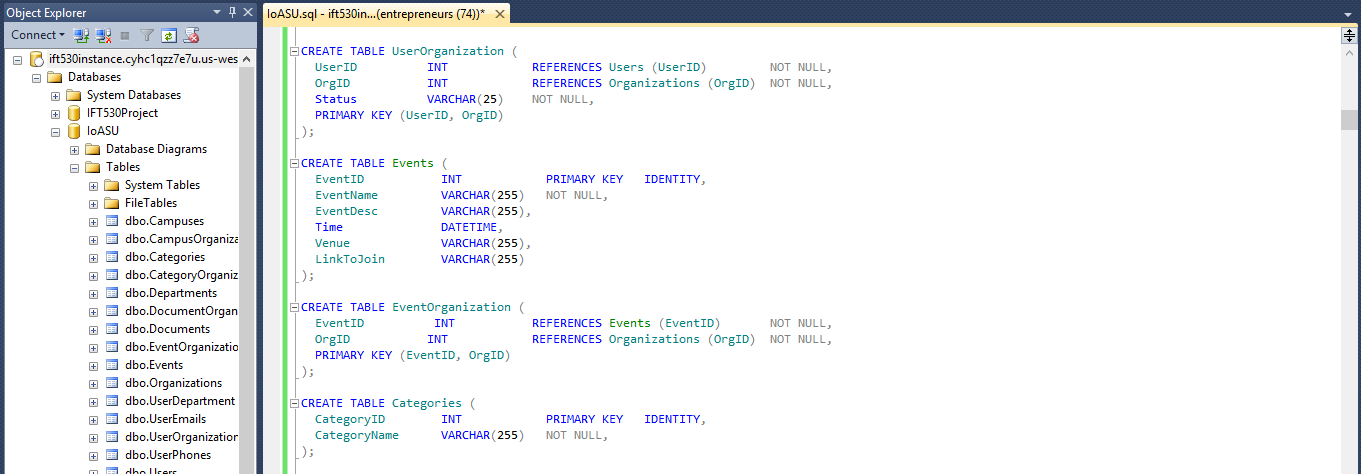
RoleID INT REFERENCES Roles (RoleID) NOT NULL,

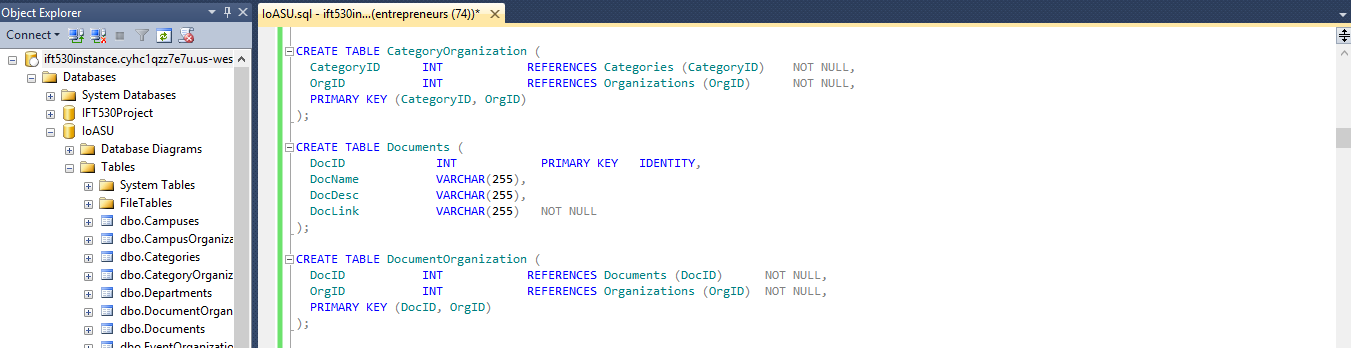
UserID INT REFERENCES Users (UserID) NOT NULL

PRIMARY KEY (RoleID, UserID)

);







**DML (Data Manipulation Language for Data Insertion)**

Sample INSERT commands are shown below. (Note: These DO NOT constitute the whole database dump. That will be attached as a separate SQL script file along with this report)

-- Insert data into the tables

SET IDENTITY\_INSERT Campuses ON;

INSERT INTO Campuses (CampusID, CampusName, Location) VALUES

(1, 'Tempe', 'Tempe'),

(2, 'Downtown', 'Phoenix'),

(3, 'Polytechnic','Mesa'),

(6, 'Thunderbird', 'Thunderbird');

SET IDENTITY\_INSERT Campuses OFF;

SET IDENTITY\_INSERT Organizations ON;

INSERT INTO Organizations (OrgID, Name, Email, Phone, MeetingVenue, WeeklyMeetDay, WeeklyMeetTime, WebLink, DateOfCreation) VALUES

(1, 'Action for America', 'aofa@asu.edu', '4806567865', 'COWDN', 'Monday', '17:00', 'aofa.org', '2012-04-01'),

(2, 'Active Minds at Arizona State University', 'aminds@asu.edu', '4805782052', 'Art', 'Friday', '16:00', 'aminds.com', '2013-05-23'),

(3, 'Adworks', 'adworks@asu.edu', '4801234357', 'ECE', 'Tuesday', '17:00', 'adworks.com', '2011-01-21'),

(15,'TECH Devils', 'techdevils@asu.edu', '3349877823', 'Fulton Center', 'Wednesday', '16:50', 'techdevils.com', '2013-11-01');

SET IDENTITY\_INSERT Organizations OFF;

INSERT INTO CampusOrganization (CampusID, OrgID) VALUES

(1, 1),

(2, 2),

(3, 3),

(6, 15);

SET IDENTITY\_INSERT Departments ON;

INSERT INTO Departments (DepID, Department) VALUES

(100, 'Information Technology'),

(200, 'Civil Engineering'),

(300, 'Biomedical Engineering'),

(400, 'Biodesign Engineering'),

(1000, 'Business Administration');

SET IDENTITY\_INSERT Departments OFF;

SET IDENTITY\_INSERT Users ON;

SELECT\* FROM Organizations;

SELECT\* FROM USERS;

INSERT INTO Users (UserID, ASUID, Password, PasswordSalt, LName, FName, Bio) VALUES

(1, 'pbuffet', HASHBYTES('SHA2\_512', 'January01@' + '1pbuffet'), '1pbuffet', 'Buffet', 'Pheobe', 'Information Technology Student'),

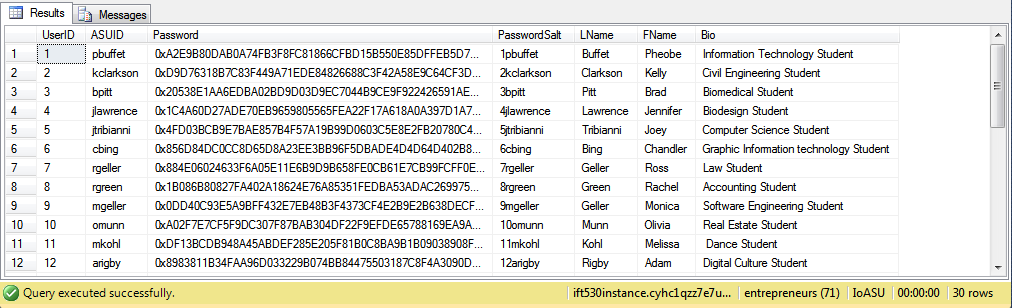
(2, 'kclarkson', HASHBYTES('SHA2\_512', 'February8' + '2kclarkson'), '2kclarkson', 'Clarkson', 'Kelly', 'Civil Engineering Student'),

(30,'jdepp', HASHBYTES('SHA2\_512', 'pirates876!' + '30jdepp'), '30jdepp', 'Depp', 'Johnny', 'Theatre Student');

-- Ideally, PasswordSalt should be a true random string for each entry.

-- For our Proof-Of-Concept purposes, a concatenation of UserID and ASUID (Both of which are unique) will work.

-- Used SHA2\_512 with a concatenation of Password and a unique PasswordSalt for each user to generate a 20-byte Encrypted Password in binary (seen as a HEX value)



SET IDENTITY\_INSERT Users OFF;

INSERT INTO UserPhones (UserID, Phone) VALUES

(1, '4806742876'),

(2, '6027321879'),

(30, '2230975234');

INSERT INTO UserEmails(UserID, Email) VALUES

(1, 'pbuffet@asu.edu'),

(2, 'kclarkson@asu.edu'),

(30, 'jdepp@asu.edu');

INSERT INTO UserDepartment (UserID, DepID) VALUES

(1, '100'),

(2, '200'),

(30, '1000');

INSERT INTO UserOrganization ( UserID, OrgID, Status) VALUES

(1, 3, 'Active'),

(25, 8, 'Active'),

(30, 1, 'Active');

SET IDENTITY\_INSERT Events ON;

INSERT INTO Events (EventID , EventName, EventDesc, Time, Venue, LinkToJoin) VALUES

(1, 'Doctoral Recital Series', 'Alpha Chi Omega presents collaborative piano recital', '17:30', 'COWDN', ''),

(2, 'Latin Sol Dance Festival', 'The event is all-day latin dance extravaganza and free for both ASU students and members of the local dance community', '18:00', 'Art', ''),

(13, 'Cutural Festival', 'Come enjoy live performances, activities and food that is representative of the variety of cultures our students bring to the ASU community.', '10:00', 'WellsFargo Arena', ''),

(14, 'Ditch the Dumpster', 'donate and recycle unwanted items as they are moving out of their residence hall ', '9:00', 'Palo Verde Residence Hall', ''),

(15, 'Vault Gallery Photo Exhibit', 'The Downtown Phoenix campus library is pleased to exhibit the work of Ryan Carey', '16:00', 'Mercado E', '');

SET IDENTITY\_INSERT Events OFF;

INSERT INTO EventOrganization(EventID, OrgID) VALUES

(1, 5),

(2, 10),

(15,15);

SET IDENTITY\_INSERT Categories ON;

INSERT INTO Categories (CategoryID, CategoryName) VALUES

(1, 'Academic'),

(2, 'Biotechnology'),

(3, 'Community and Lifestyle'),

(4, 'Data and Analytics'),

(5, 'Entrepreneurship/Innovation'),

(6, 'Financial Services'),

(7, 'International'),

(8, 'Political'),

(9, 'Science and Engineering'),

(10,'Sports'),

(11,'Sustainability');

SET IDENTITY\_INSERT Categories OFF;

INSERT INTO CategoryOrganization (CategoryID , OrgID) VALUES

(1, 1),

(11, 1);

SET IDENTITY\_INSERT Documents ON;

INSERT INTO Documents (DocID, DocName, DocDesc, DocLink) VALUES

(1, 'Member List', 'List of all members of the organization', 'aminds.com/memberlist'),

(2, 'Student Event Planner', 'Event Planning Manual', 'adworks.com/studenteventplanner'),

(10, 'Meeting Presentation', 'Presentations for events ', 'biosyntagma.org/presentation1');

SET IDENTITY\_INSERT Documents OFF;

INSERT INTO DocumentOrganization (DocID, OrgID) VALUES

(1,2),

(2,3),

(3,6),

(4,9),

(5,10),

(6,7),

(7,9),

(8,6),

(9,4),

(10,8);

INSERT INTO Roles (RoleID, Description) VALUES

(1, 'SuperAdmin'),

(2, 'OrgAdmin'),

(3, 'User');

INSERT INTO UserRoles (RoleID, UserID) VALUES

('1', '1'),

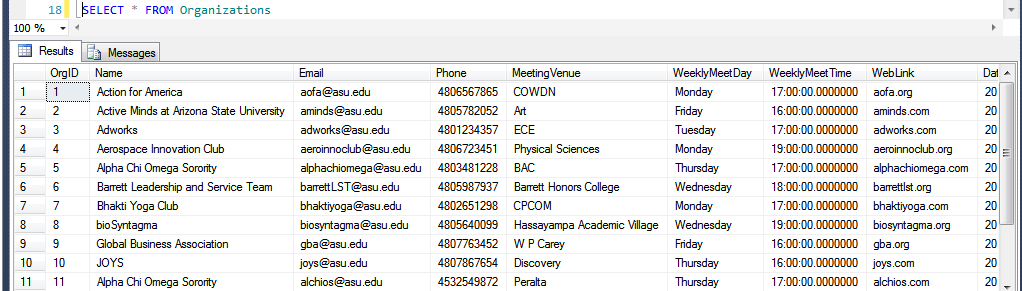
('2', '5'),

('2', '7'),

('3', '12'),

('3', '19'),

('3', '25');



**SQL Queries**

**Views**

Some of the views used to display table data are shown below.

USE IoASU;

--View for campuses and number of organizations on each campus

GO

create view CampusOrganizationCountView as

select Campuses.CampusID,CampusName,count(\*) as 'Number of organizations'

from Campuses inner join CampusOrganization

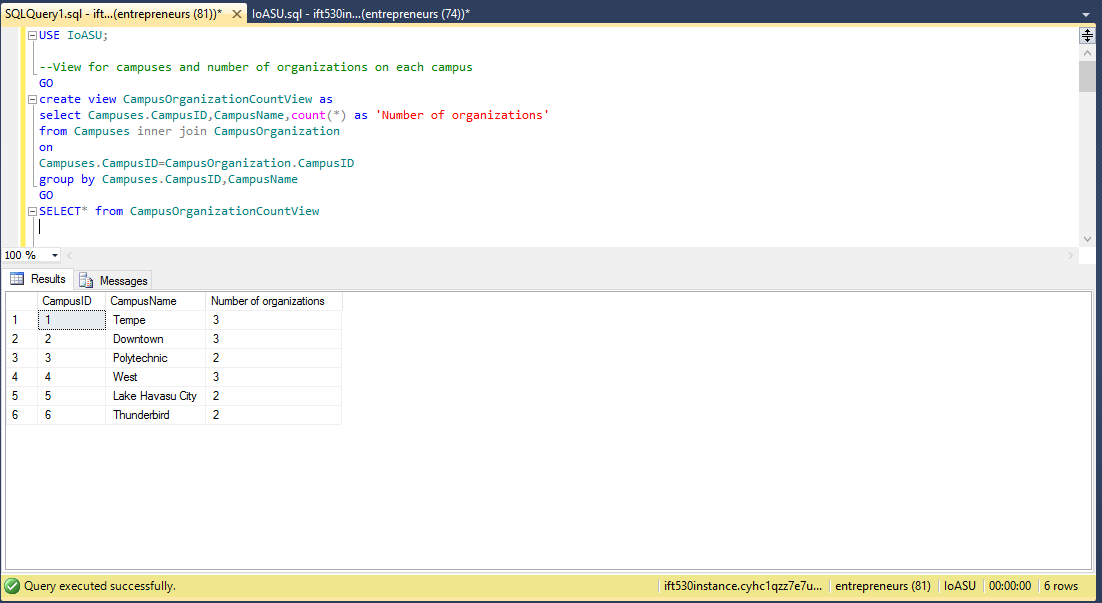
on

Campuses.CampusID=CampusOrganization.CampusID

group by Campuses.CampusID,CampusName

GO

SELECT\* from CampusOrganizationCountView



--View for categories and number of organizations

GO

create view CampusOrganizationCountView as

select Categories.CategoryID,CategoryName,count(\*) as 'Number of organizations'

from Categories inner join CategoryOrganization

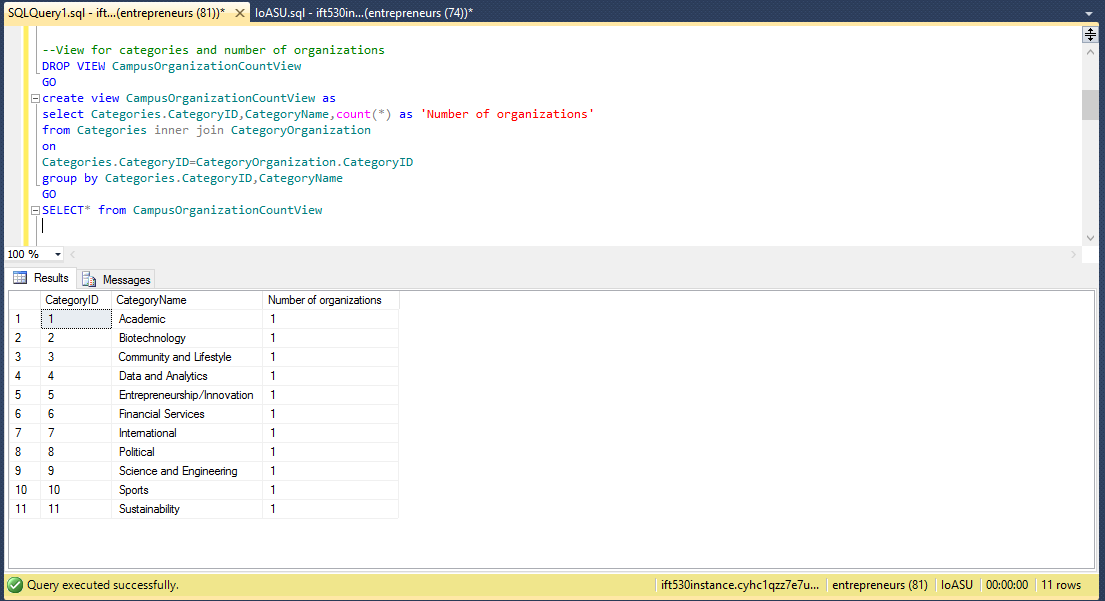
on

Categories.CategoryID=CategoryOrganization.CategoryID

group by Categories.CategoryID,CategoryName

GO

SELECT\* from CampusOrganizationCountView



Stored Procedures:

All the stored procedures were developed to be used to fetch and return data for the mobile application’s different pages.

-- Stored Procedure:

-- Return list of Organizations (Name, Email, Weblink) given a UserID

USE IoASU;

GO

CREATE PROC sp\_OrgListForUser

@UserID INT

AS

BEGIN

SELECT Name, Email, WebLink

FROM Organizations WHERE OrgID IN

(SELECT OrgID

FROM UserOrganization WHERE UserID = @UserID)

END;

-- Test:

-- EXEC sp\_OrgListForUser 1

-- Stored Procedure:

-- Return stuff for User's My Profile Page

SELECT \* FROM UserPhones

GO

CREATE PROC sp\_MyProfileForUser

@UserID INT

AS

BEGIN

SELECT Fname, Lname, ASUID, d.Department, Bio, Email, Phone

FROM Users u

JOIN UserDepartment ud ON u.UserID = ud.UserID

JOIN Departments d ON d.DepID = ud.DepID

JOIN UserEmails ue ON u.UserID = ue.UserID

JOIN UserPhones up ON u.UserID = up.UserID

WHERE u.UserID = @UserID

END;

-- Test:

-- sp\_MyProfileForUser 1

-------------------------------------------------------------------------------------------------

-- Procedure to Validate User.

-- Three cases or output possibilities:

-- 1. 'NA' = User Does Not Exist

-- 2. 'Success' = Login Successful

-- 3. 'Failure' = Password Incorrect

GO

CREATE PROC sp\_ValidateUser

@ASUID VARCHAR(20),

@Password VARCHAR(20)

AS

DECLARE @PasswordSalt VARCHAR(128);

DECLARE @Result VARCHAR(20);

IF (SELECT Password from Users WHERE ASUID = @ASUID) IS NULL

SET @Result = 'NA'

ELSE

BEGIN

IF (SELECT HASHBYTES('SHA2\_512', @Password + PasswordSalt) FROM Users WHERE ASUID = @ASUID) =

(SELECT Password FROM Users WHERE ASUID = @ASUID)

SET @Result = 'Success'

ELSE SET @Result = 'Failure'

;END

SELECT @Result;

-------------------------------------------------------------------------------------------------

/\* Testing:

-- USE as Dynamic SQL

DECLARE @ASUID VARCHAR(20);

DECLARE @Password VARCHAR(20);

DECLARE @SQLquery NVARCHAR(50);

SET @ASUID = 'tcruise'

SET @Password = 'topgun&&'

-- Implement below in login() function of Python in Django

SET @SQLQUERY = N'sp\_ValidateUser ' + '''' + @ASUID + ''', ' + '''' + @Password + ''''

EXEC sp\_executesql @SQLQUERY

\*/

**Cursors**

To make the application engage users personally, a cursor was developed which generates a personalized email or notification templates for each user based on the organization(s) which she has registered to.

-- Emails Cursor

-- Description: Given a particular organization, this cursor

-- sends an email to all the users one-by-one

-- about the latest event in the organization.

DECLARE @OrgID INT,

@currentEmail VARCHAR(255),

@FirstName VARCHAR(255),

@LastName VARCHAR(255),

@EventName VARCHAR(255),

@EventDesc VARCHAR(255),

@EventTime VARCHAR(255),

@EventVenue VARCHAR(255),

@hello VARCHAR(100),

@subject VARCHAR(255),

@message VARCHAR(1000);

PRINT '-------- User Emails --------';

-- Pass OrgID dynamically here.

SET @OrgID = 1;

SELECT @EventName = EventName, @EventDesc = EventDesc,

@EventTime = Time, @EventVenue = Venue

FROM Events WHERE EventID IN

(SELECT TOP 1 EventID FROM EventOrganization WHERE OrgID = @OrgID);

DECLARE emails\_cursor CURSOR FOR

SELECT Email, u.FName, LName FROM UserEmails ue

JOIN Users u ON ue.UserID = u.UserID

WHERE ue.UserID IN

(SELECT UserID FROM UserOrganization WHERE OrgID = @OrgID);

OPEN emails\_cursor

FETCH NEXT FROM emails\_cursor

INTO @currentEmail, @FirstName, @LastName

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT ' '

SELECT @hello = 'Hi ' + @FirstName + ' ' + @LastName + ',' + CHAR(13)+CHAR(10)

SELECT @subject = 'Subject: ' + @EventName + CHAR(13)+CHAR(10)

SELECT @message = 'You are invited to this event! Details: ' + @EventDesc + CHAR(13)+CHAR(10)

+ 'Time: ' + @EventTime + CHAR(13)+CHAR(10)

+ 'Venue: ' + @EventVenue + CHAR(13)+CHAR(10)

+ 'Hope to see you there!' + CHAR(13)+CHAR(10)

+ 'Bye.'

PRINT '<'+@currentEmail+'>' + CHAR(13)+CHAR(10)+ @hello + @subject + @message

-- Get the next user.

FETCH NEXT FROM emails\_cursor

INTO @currentEmail, @FirstName, @LastName

END

CLOSE emails\_cursor;

DEALLOCATE emails\_cursor;

Sample output:

-------- User Emails --------

<omunn@asu.edu>

Hi Olivia Munn,

Subject: Ignite @ ASU

You are invited to this event! Details: Public event for individuals to share the ideas, passions and stories

Time: Jan 1 2017 1:00PM

Venue: CPCOM

Hope to see you there!

Bye.

<mkohl@asu.edu>

Hi Melissa Kohl,

Subject: Ignite @ ASU

You are invited to this event! Details: Public event for individuals to share the ideas, passions and stories

Time: Jan 1 2017 1:00PM

Venue: CPCOM

Hope to see you there!

Bye.

<sjohansson@asu.edu>

Hi Scarlett Johansson,

Subject: Ignite @ ASU

You are invited to this event! Details: Public event for individuals to share the ideas, passions and stories

Time: Jan 1 2917 1:00PM

Venue: CPCOM

Hope to see you there!

Bye.

**Triggers**

/\*

Trigger to drop document from Documents table when

the record is deleted from DocumentOrganization table.

\*/

GO

CREATE TRIGGER DocumentDropTrigger

ON DocumentOrganization

FOR DELETE AS

BEGIN

DELETE FROM Documents WHERE DocID=(SELECT DocID FROM deleted)

END;

**Indexing and Dynamic SQL**

Dynamic SQL was used for the ‘Search’ functionality using which users search for organizations they wish to join to. To make the querying fetch results faster and dynamically (just as the user types in to the search bar, and without even having to press Search after typing), we used a non-clustered index on the Name column of the Organizations table.

------------------

-- Create a non-clustered index on Organizations table for Name field

GO

-- Find an existing index named IX\_Orgs\_Name and delete it if found.

IF EXISTS (SELECT name FROM sys.indexes

WHERE name = N'IX\_Orgs\_Name')

DROP INDEX IX\_Orgs\_Name ON Organizations;

GO

-- Create a nonclustered index called IX\_Orgs\_Name

-- on the Organizations table using the Name column.

CREATE NONCLUSTERED INDEX IX\_Orgs\_Name

ON Organizations (Name);

GO

--------------------

-- This procedure implements the Organization Search function

-- for users

GO

CREATE PROC sp\_SearchOrgs (@SearchTerm NVARCHAR(max) = '')

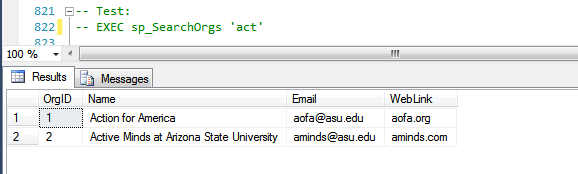
AS

DECLARE @cmd NVARCHAR(max);

SET @cmd = N'SELECT OrgID, Name, Email, WebLink FROM Organizations WHERE Name LIKE ''%' + @SearchTerm + '%''' + ' OR EMAIL LIKE ''%' + @SearchTerm + '%''';

EXEC sp\_executesql @cmd

GO



**Data Control Language (DCL)**

For separation of scope, we have two types of users, as below.

CREATE LOGIN SuperAdmin WITH PASSWORD = '8SuNy<76G,',

DEFAULT\_DATABASE = IoASU,

CHECK\_EXPIRATION =ON,

CHECK\_POLICY =ON;

CREATE USER SuperAdmin;

-- SuperAdmin can perform any activity on IoASU Database

GRANT ALTER ON DATABASE::IoASU

TO SuperAdmin;

---------------------------------------------------------------------------

-- General User, who uses the mobile application.

-- This user has permissions to view, insert and update tables in dbo schema.

CREATE LOGIN JustAnotherUser WITH PASSWORD = '<AK31r-B8C',

DEFAULT\_DATABASE = IoASU,

CHECK\_EXPIRATION =ON,

CHECK\_POLICY =ON;

CREATE USER JustAnotherUser;

GRANT SELECT, UPDATE, INSERT

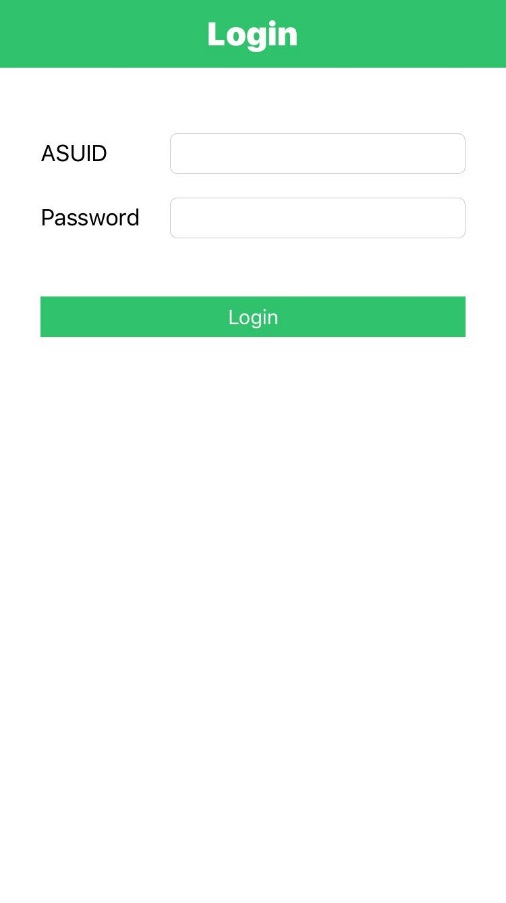
ON SCHEMA::dbo

TO JustAnotherUser;

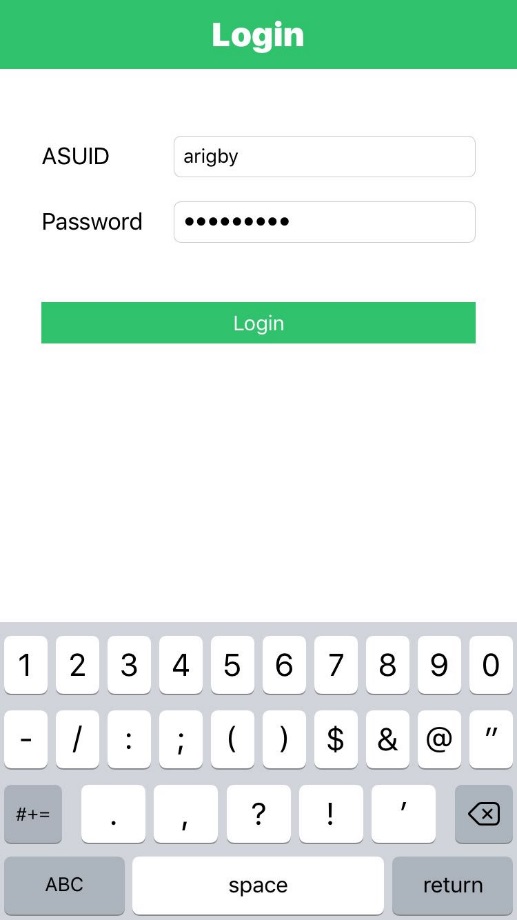
**Front-End (iOS application)**

The full functioning of the app will be shown live on demo day. Here are a couple of screenshots of the app’s working.

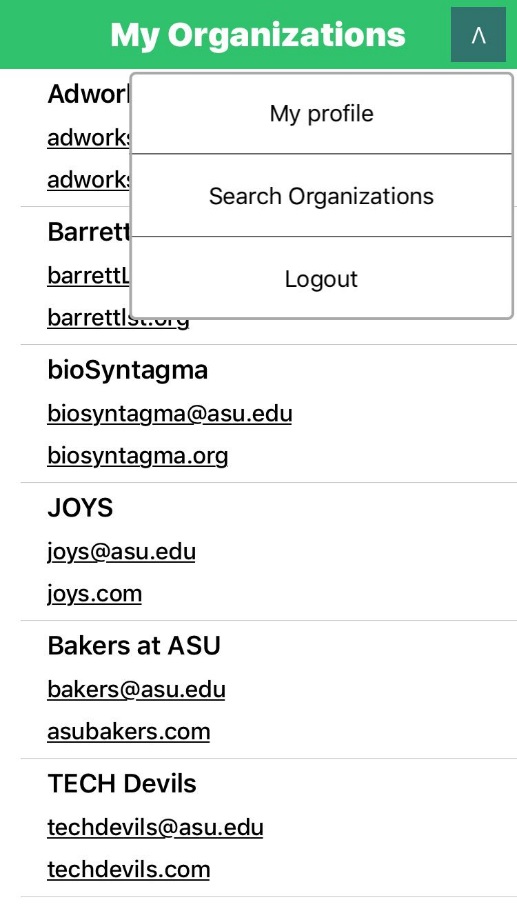
The login screen:



Login as a user:



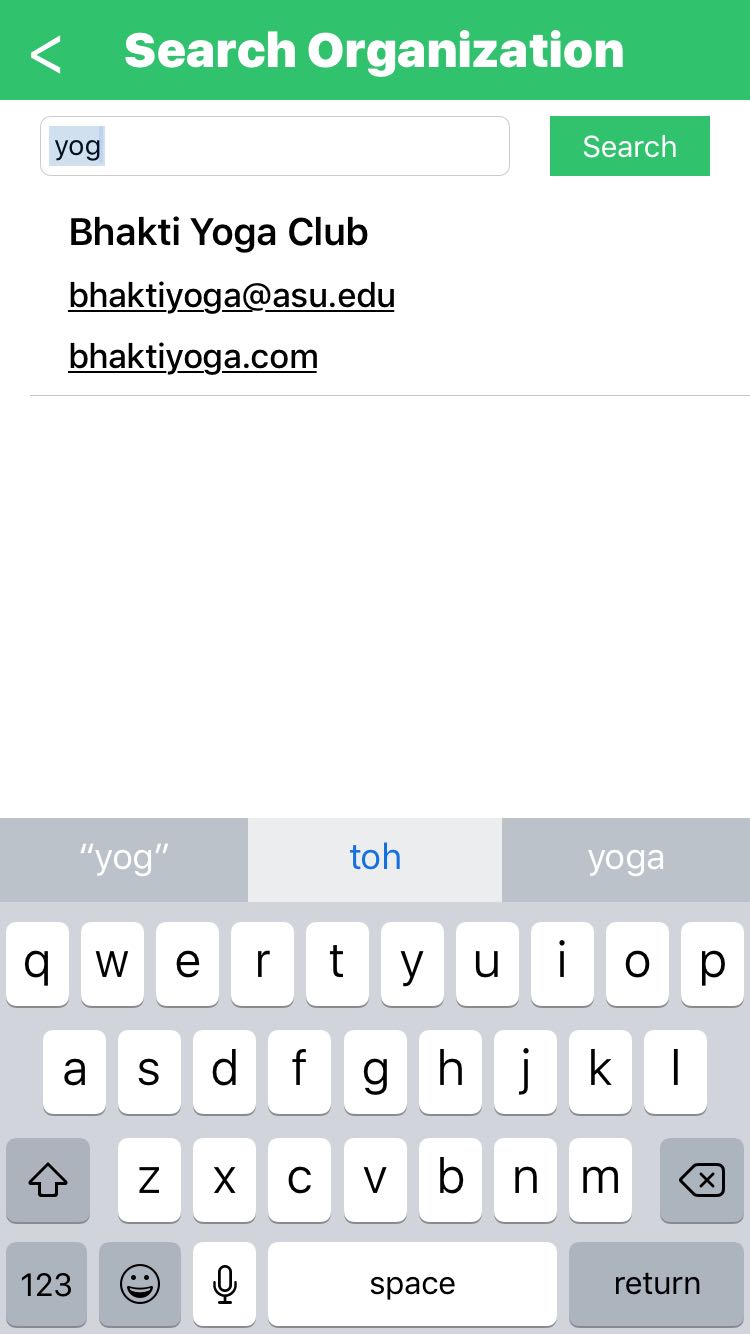
List of organizations in which the user is registered in:



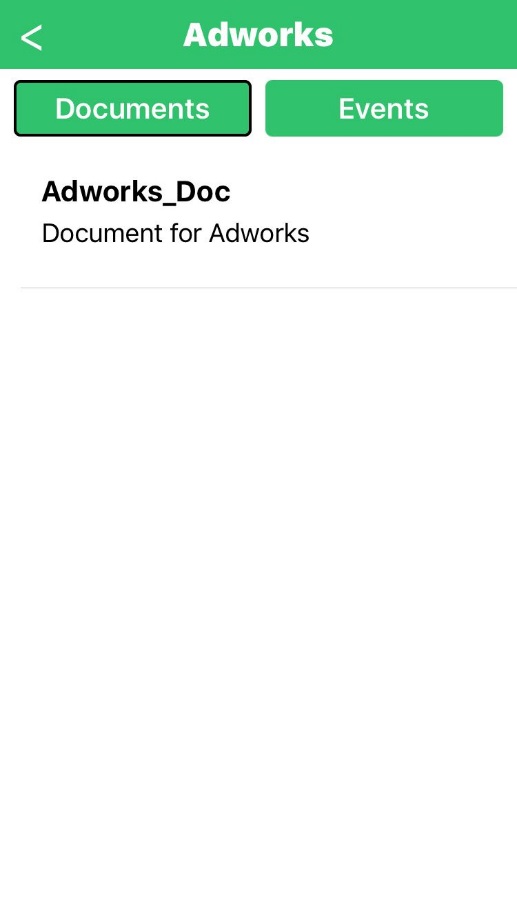
The user’s profile:



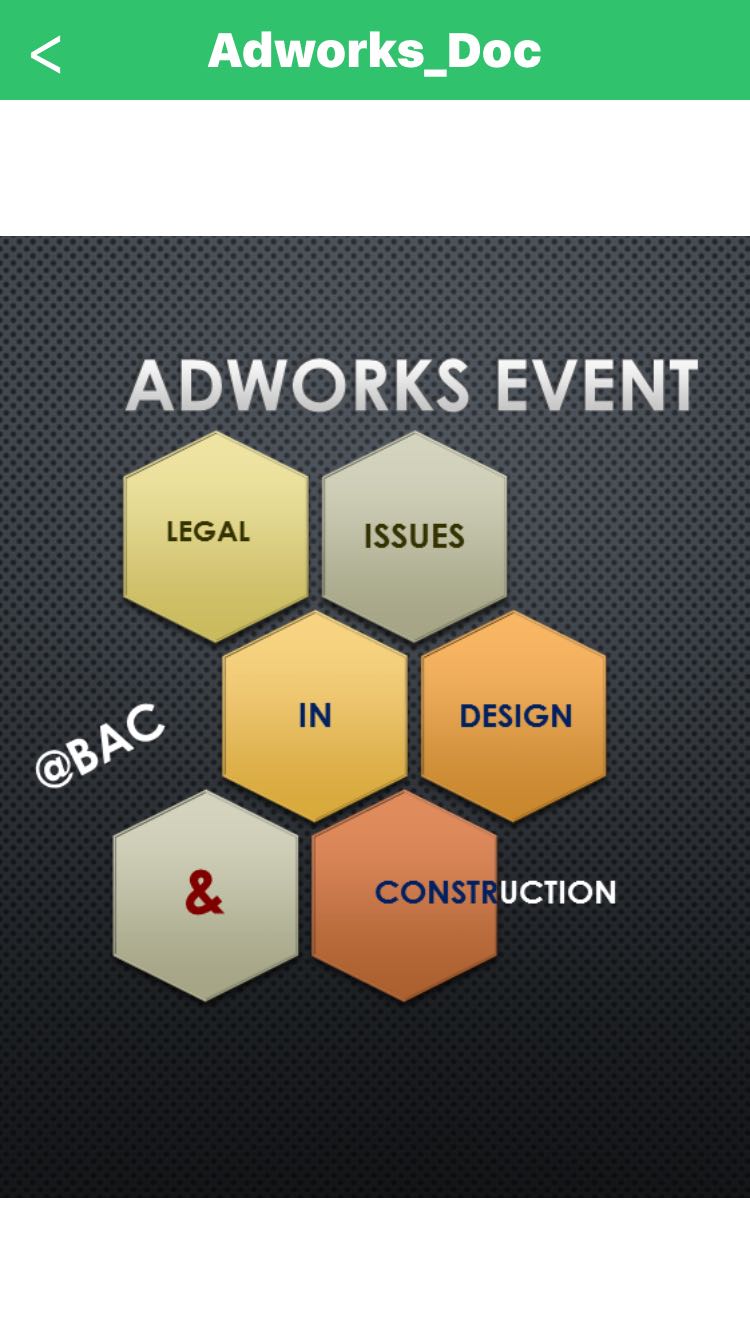
Search Organizations:



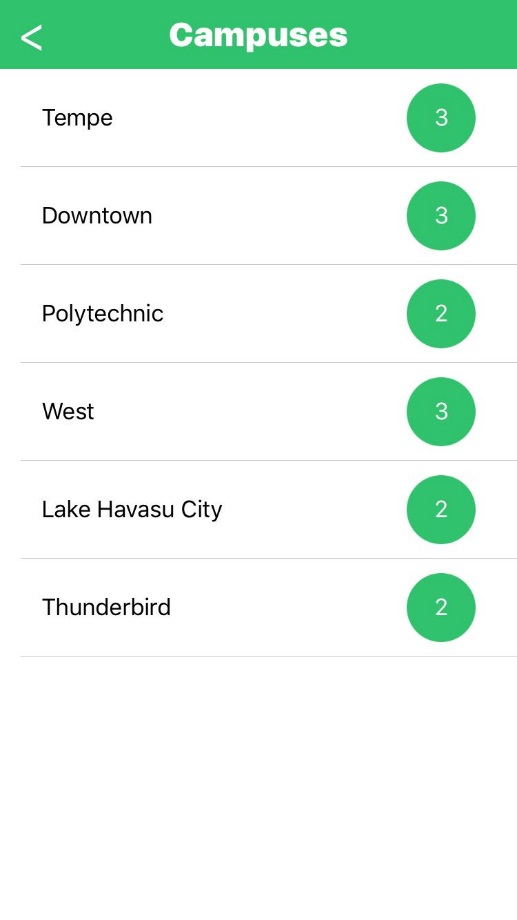
Data of an organization:



Document (fetched from AWS S3 bucket):



Super Admin page:



**Conclusion and Future Work**

With the database and its objects in place; we went about developing a working front-end (a mobile app, which will be demoed on Presentation day). We have established an end to end connectivity, i.e. from Database to the User.

A couple more features could be added to the database, like an Intelligent Recommendation System for users (recommend Organizations the user most probably be interested in, given a set of data points like current memberships, other students’ memberships in the same network, etc).

**References**

Murach's SQL Server 2016 for Developers (by Joel Murach, Bryan Syverson)

Transact-SQL Reference (Database Engine) | Microsoft Docs (<https://docs.microsoft.com/en-us/sql/t-sql/language-reference>)

Database Administrators Stack Exchange (<https://dba.stackexchange.com/>)

**Database dump: **