CS353
DATABASE SYSTEMS

Design Report

Project Topic: **Social Network For Check-Ins**

Project Name:**'WhereTo'**

Project TA: Fuat Basık

Link to webpage: https://github.com/FaaizHaque/CS353-Group29

A Project By Group 29

Hamza Saeed Khan - 21500570 - Section 3

Faaiz Ul Haque - 21503527 - Section 3

Usama Saqib - 21500116 - Section 2

Askari Iqbal - 21504228 - Section 3

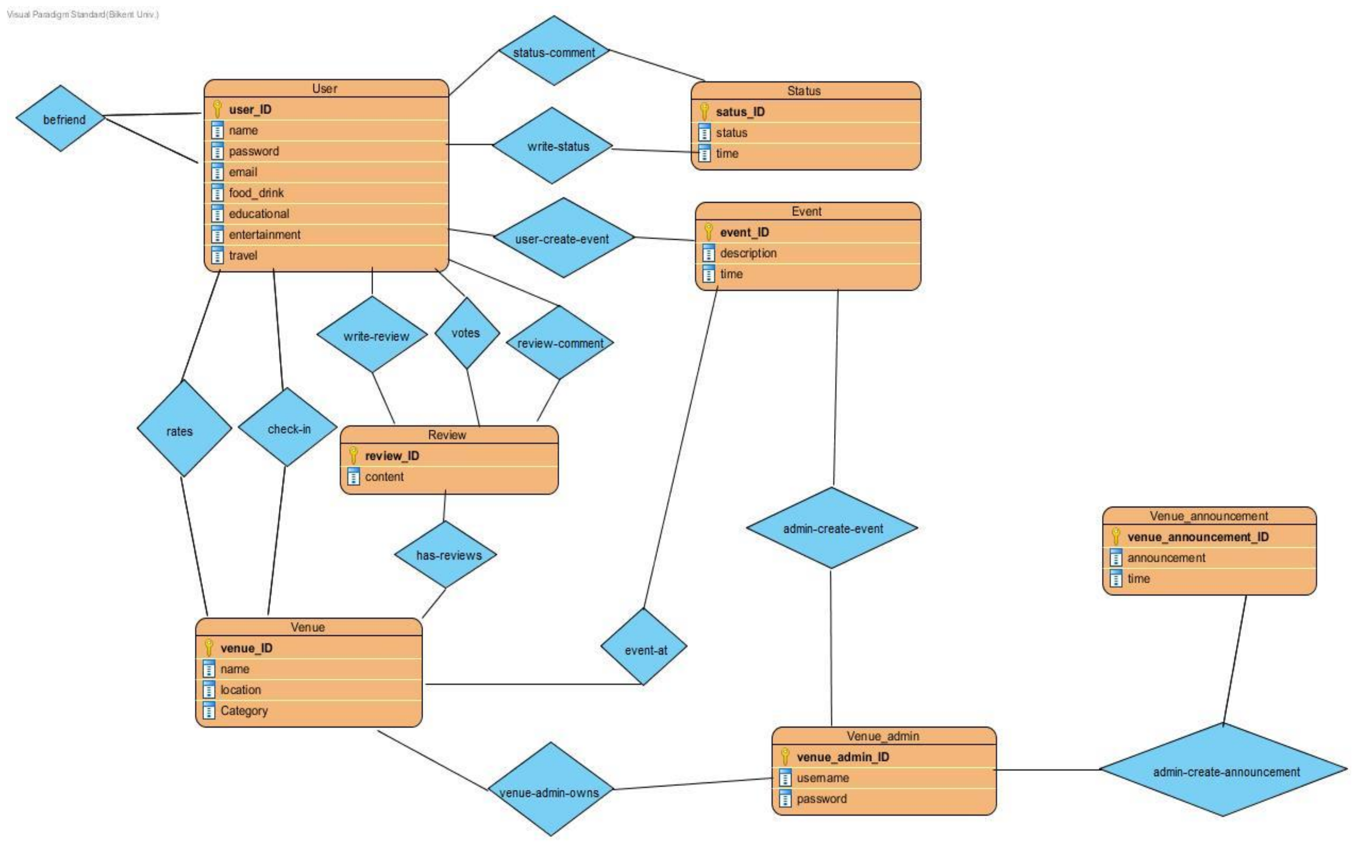
Contents

1. Introduction

This is our design report for our project called ‘WhereTo’ which is a social network application for user check-ins. This report will show our revised E-R diagram from our first report along with its relational schema with normalization of BCNF. In our software design specifications we have also included the functional components, the user interface design along with its corresponding SQL statements and finally the advanced database components.

2. Revised E-R Model

We received some feedback regarding our initial E-R model stating the overall design seemed to simple, and required more entities. We added the following entities to our original diagram: Status, Event, Venue_Announcement, Venue_Admin and Review. Furthermore, we created many more relations to further define these entities and how they interact with each other. We also had incorrect notation of many-to-one, one-to-one, and many-many entities and also improved on those. We added weak-entities and total participation where applicable.



3. Relational Schema

1 **User**(<u>user_id,</u> name , password, email, food_drink, educational, entertainment, travel)
Domain of name must have max of 30 characters and not null
Domain of 'password' must have a minimum of 8 characters
'user_id' is a candidate key and the selected primary key

2 **Venue**(<u>venue_id,</u> name, location, category)

Domain of ‘category’ has only four options, as listed in user entity’s attributes (food_drink, educational, entertainment, travel)
‘venue_id’ is a candidate key and the selected primary key

3 **Review**(review_id, content)
‘review_id’ is a candidate key and the selected primary key

4 **Status**(status_id, status, time)
Domain of ‘time’ is standard to the 24 hour clock (i.e (0-23):(00-60))
‘status_id’ is a candidate key and the selected primary key

5 **Event**(event_id, description, time)
Domain of ‘time’ is standard to the 24 hour clock (i.e (0-23):(00-60))
‘event_id’ is a candidate key and the selected primary key

6 **Venue_Admin**(venue_admin_id, username, password)
Domain of ‘password’ must have a minimum of 8 characters
‘venue_admin_id’ is a candidate key and the selected primary key

7 **Venue_Announcement**(venue_announcement_id, announcement, time)
Domain of ‘time’ is standard to the 24 hour clock (i.e (0-23):(00-60))
‘venue_announcement_id’ is a candidate key and the selected primary key

8 **befriend**(friend_ID_1, friend_ID_2, friendship_status)
Domain of ‘friendship_status’ is either friends or not friends (0 or 1 or bool)
Primary key is friend_ID_1 and friend_ID_2 combined
FK: ‘friend_ID_1’ referencing User (user_ID)
FK: ‘friend_ID_2’ referencing User (user_ID)

9 **check-in**(user_ID, venue_ID , time)
Domain of ‘time’ is standard to the 24 hour clock (i.e (0-23):(00-60))
Primary key is user_ID and venue_ID combined
FK: ‘user_ID’ referencing User
FK: ‘venue_ID’ referencing venue

10 **write-status**(user_ID, status_ID)
Primary key is a combination of user_ID and status_ID
FK: ‘user_ID’ referencing User
FK: ‘status_ID’ referencing Status

11 **status-comment**(status_ID, user_ID, content, time)
Domain of ‘time’ is standard to the 24 hour clock (i.e (0-23):(00-60))
FK: ‘user_ID’ referencing User
FK: ‘status_ID’ referencing Status

12 **review-comment**(review_ID, user_ID, content, time)
Domain of ‘time’ is standard to the 24 hour clock (i.e (0-23):(00-60))
Primary key is review_ID and user_ID together
FK: ‘user_ID’ referencing User
FK: ‘review_ID’ referencing Review

13 **votes**(review_ID, user_ID, vote)
Domain of ‘vote’ is 1 or -1 (upvote or downvote)
Primary key must be a combination of all attributes since review-comment has the primary key of review_ID + user_ID
FK: ‘user_ID’ referencing User
FK: ‘review_ID’ referencing Review

14 **has-reviews**(venue_ID, review_ID)
Primary key is venue_ID + review_ID
FK: ‘venue_ID’ referencing Venue
FK: ‘review_ID’ referencing Review

15 **rates**(user_ID, venue_ID, rating)
Domain of ‘rating’ is from 0 to 5 (stars)
Primary key: user_ID + venue_ID + rating
FK: ‘user_ID’ referencing User
FK: ‘venue_ID’ referencing Venue

16 **write-review**(writers_ID, review_ID)
Primary key: writers_ID + review_ID
FK: ‘writers_ID’ referencing User (user_ID)
FK: ‘review_ID’ referencing Review

17 **user-creates-event**(event_ID, user_ID)
Primary key: event_ID + user_ID
FK: ‘event_ID’ referencing Event
FK: ‘user_ID’ referencing User

18 **admin-creates-event**(venue_ID, venue_announcement_ID)
Primary key: venue_ID + venue_announcement_ID
FK: ‘venue_ID’ referencing Venue
FK: ‘venue_announcement_ID’ referencing Venue_Announcement

19 **event-at**(event_ID, venue_ID)
Primary Key: event_ID + venue_ID
FK: ‘event_ID’ referencing Event
FK: ‘venue_ID’ referencing Venue

20 **venue-admin-owns**(<u>venue_admin_ID, venue_ID</u>)
Primary Key: venue_admin_ID + venue_ID
FK: ‘venue_ID’ referencing Venue
FK: ‘venue_admin_ID’ referencing Venue_Admin

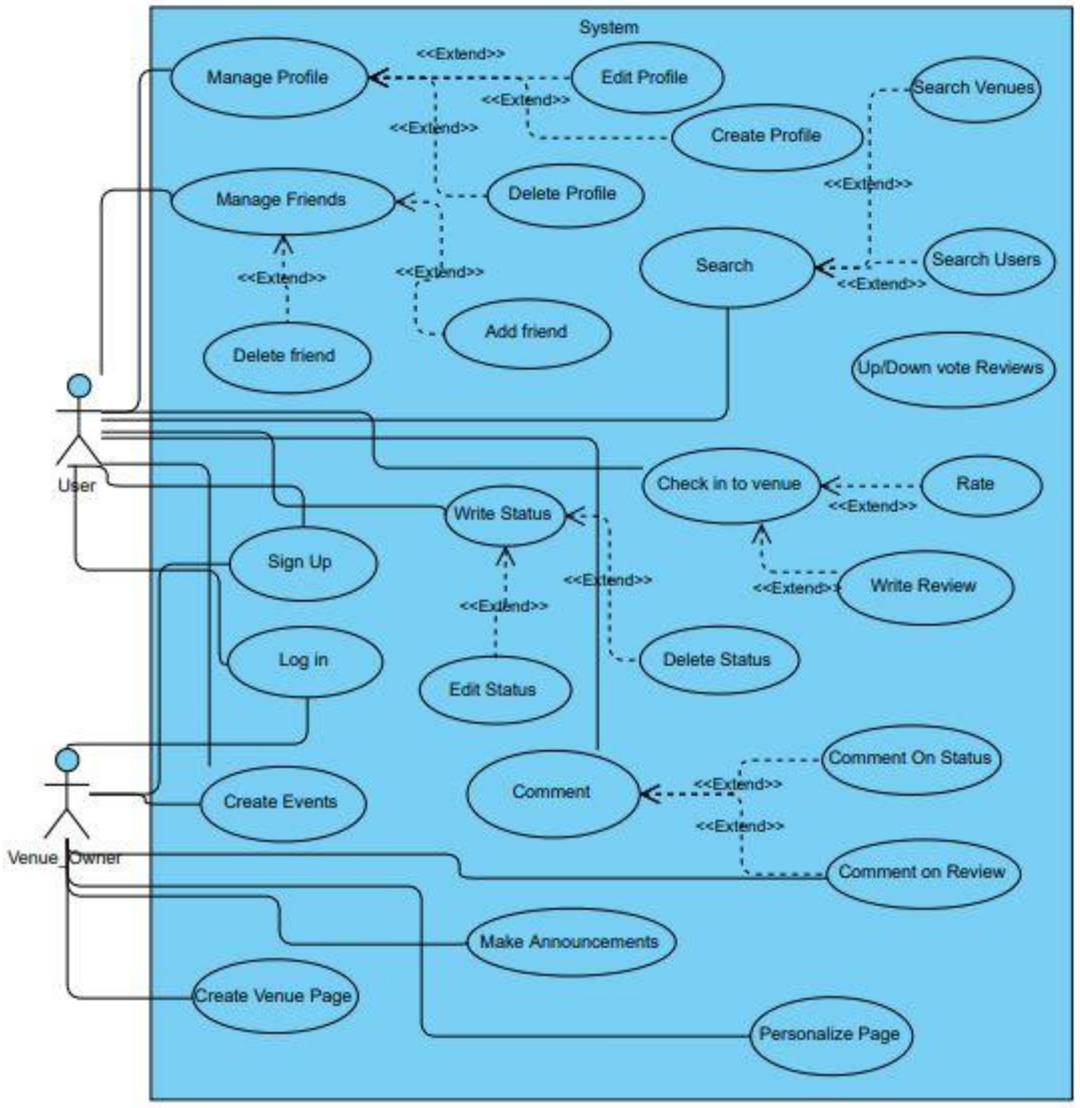
21 **admin-create-announcement**(<u>venue_admin_ID, venue_announcemnet_ID</u>)
Primary Key: venue_admin_ID + venue_announcement_ID
FK: ‘venue_admin_ID’ referencing Venue_Admin
FK: ‘venue_announcement_ID’ referencing Venue_Announcement

4. Table Creations

```
Create Table User (
        user_id         int,
        name            varchar(30),
        password        varchar(30),
        email           varchar(30),
        food_drink      bool,
        educational     bool,
        entertainmentbool,
        travel          bool,
        PRIMARY KEY(user_id)
        );

Create Table Venue (
        venue_id        int,
        name            varchar(30),
        location        varchar(50),
        category        varchar(20),
        PRIMARY KEY(venue_id)
        );

Create Table Review (
        review_id       int,
        content         varchar(300),
        PRIMARY KEY(review_id)
        );

Create Table Status (
        status_id       int,
        status          varchar(300),
        time            datetime()
        PRIMARY KEY(status_id)
        );

Create Table Event (
        event_id        int,
        description     varchar(300)
        time    datetime(),
        PRIMARY KEY(event_id)
        );

Create Table Venue_Admin(
        venue_admin_id          int,
        username                varchar(30),
        password                varchar(30),
        PRIMARY KEY(venue_admin_id)
        );

Create Table Venue_Announcement(
        venue_announcement_id       int,
        announcement                varchar(300),
        time                        datetime(),
        PRIMARY KEY(venue_announcement_id)
        );

Create Table befriend(
        friend_ID_1     int,
        friend_ID_2     int,
        friendship_status bool,
        PRIMARY KEY(friend_ID_1, friend_ID_2),
        FOREIGN KEY(friend_ID_1) references User,
        FOREING KEY(friend_ID_2) references User
        );

Create Table check-in(
        user_ID         int,
        venue_ID        int,
        time            datetime(),
        PRIMARY KEY(user_ID, venue_ID),
        FOREIGN KEY(user_ID) references User,
        FOREIGN KEY(venue_ID) references Venue
        );

Create Table write-status(
        user_ID         int,
        status_ID       int,
        PRIMARY KEY(user_ID, status_ID),
        FOREIGN KEY(user_ID) references User,
        FOREIGN KEY(status_ID) references Status
        );

Create Table status-comment(
        status_ID       int,
        user_ID         int,
        content         varchar(100),
        time            datetime(),
        FOREIGN KEY(user_ID) references User,
        FOREIGN KEY(status_ID) references Status
        );

Create Table review-comment(
        review_ID       int,
        user_ID         int,
        content         varchar(100),
        time            datetime(),
        PRIMARY KEY(review_ID, user_ID)
        FOREIGN KEY(user_ID) references User,
        FOREIGN KEY(review_ID) references Review
        );

Create Table votes(
        review_ID       int,
        user_ID         int,
        vote            int,
        PRIMARY_KEY(review_ID, user_ID, vote),
        FOREIGN KEY(user_ID) references User,
        FOREIGN KEY(review_ID) references Review
        );

Create Table has-reviews(
        venue_ID        int,
        review_ID       int,
        PRIMARY KEY(venue_ID, review_ID),
        FOREIGN KEY(review_ID) references Review,
        FOREIGN KEY(venue_ID) references Venue
        );

Create Table rates(
        user_ID         int,
        venue_ID        int,
        rating          int,
        PRIMARY KEY( user_ID, venue_ID, rating),
        FOREIGN KEY(user_ID) references User,
        FOREIGN KEY(venue_ID) references Venue
        );

Create Table write-review(
        writers_ID      int,
        review_ID       int,
        PRIMARY KEY( writers_ID, review_ID ),
        FOREIGN KEY(writers_ID) references User,
        FOREIGN KEY(review_ID) references Review
        );

Create Table user-creates-event(
        event_ID        int,
        user_ID         int,
        PRIMARY KEY( event_ID, user_ID)
        FOREIGN KEY(user_ID) references User,
        FOREIGN KEY(event_ID) references Event
        );


Create Table admin-creates-event(
        venue_ID                        int,
        venue_announcement_ID           int,
        PRIMARY KEY(venue_ID, venue_announcement_ID)
        FOREIGN KEY(venue_ID) references Venue,
        FOREIGN KEY(venue_announcement_ID) references Venue_Announcement
        );

Create Table event-at(
        event_ID        int,
        venue_ID        int,
        PRIMARY KEY(event_ID, venue_ID),
        FOREIGN KEY(event_ID) references Event,
        FOREIGN KEY(venue_ID) references Venue

Create Table venue-admin-owns(
        venue_admin_ID          int,
        venue_ID                int,
        PRIMARY KEY(venue_admin_ID, venue_ID),
        FOREIGN KEY(venue_admin_ID) references Venue_Admin,
        FOREIGN KEY(venue_ID) references Venue
        );

Create Table admin-create-announcement(
        venue_admin_ID          int,
        venue_announcement_ID      int,
        PRIMARY KEY(venue_admin_ID, venue_announcement_ID),
        FOREIGN KEY(venue_admin_ID) references Venue_Admin,
        FOREIGN KEY(venue_announcement_ID) references Venue_Announcement
        );
```

5. Functional Components

Use Case Diagram



Users

- Users can sign in to the program
- Log-in
- Create a personalized profile
- Search people
- Send friend requests
- Accept friend requests
- Unfriend another user
- Search for a venue

- Check-in to a venue
- Rate a venue
- Review a venue
- Upvote or downvote a review
- Write a status
- Comment on a status
- Delete a status
- Create events
- Edit their profile
- Delete their profile
- Chat with other users

Venue Owners

- Sign in to the program
- Log-in
- Create a page for their venue
- Personalize their venue page
- Make announcements that become visible on the venue page
- Create events at their own venues
- Reply to a review

Algorithms

- Before making an account, the system must ensure that a user with the same user ID does not already exist

- Before allowing a user to message another user, the system must first confirm that they are friends, and throw an error message if not.

- Before allowing a user to edit a venue page, an announcement on that page, the system must ensure that the one acting is the owner of that specific venue.

- The program must ensure that, only friends can comment on the statuses of other friends.

- The program must ensure that, statuses of users are not visible to non friends.

- The program must ensure that only the user can edit their profile, write statuses, check in to places or create event from their own profiles.

- The system should not allow a user to upvote or downvote a single review more than once.

Data Structures

The system will be designed so that the data can be held on a remote server just like any large scale web application, however, for the purposes of the demo a laptop used. SQL will be used to manage the database but to handle the data returned by SQL queries, a variety of data structures will be used base on the appropriate considerations of the need of the moment.

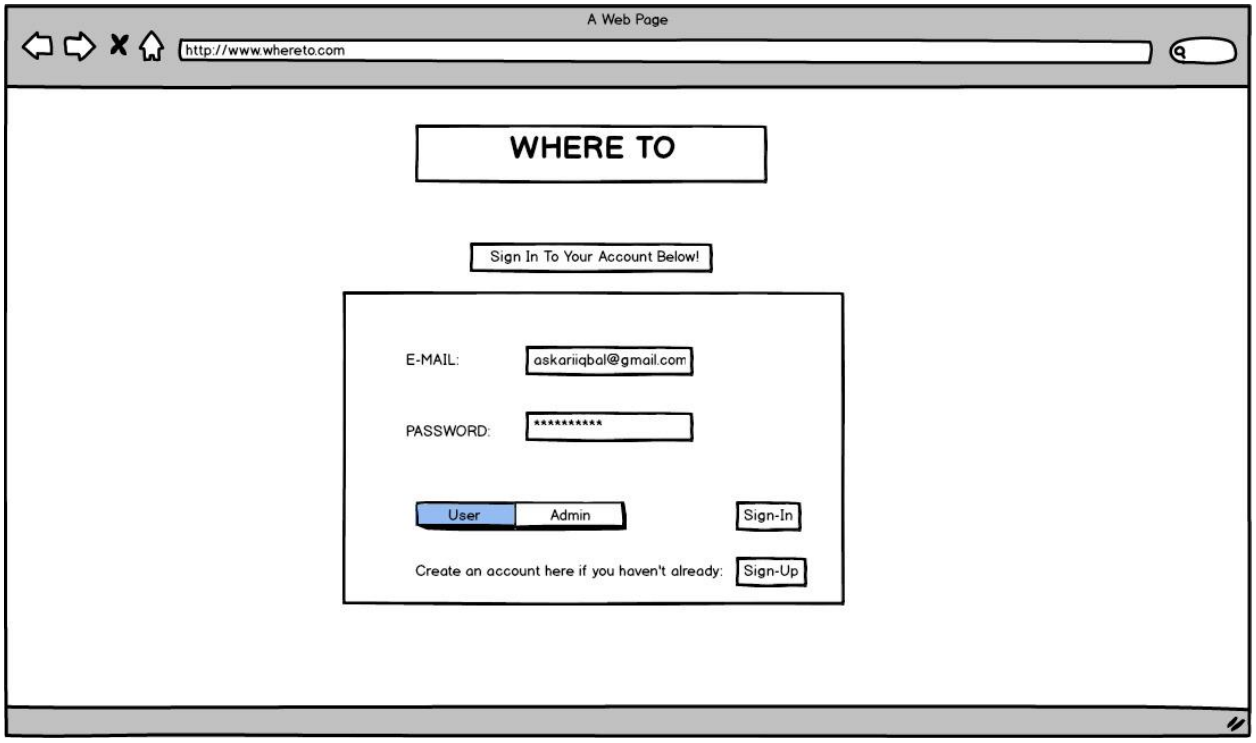
Stacks will be used for comments underneath a status. The reason being that comments will appear in reverse chronological order with the latest comments on the top.

Arrays will be used for friend lists because of constant retrieval time. Arrays will also be used for storing numbers, such as, number of upvotes and downvotes because, of the relative ease with which arrays handle numerical values.

Hash Tables will be sued for password and user name storage, to ensure an extra layer of security.

6. User Interface and SQL statements

5.1 Sign-In

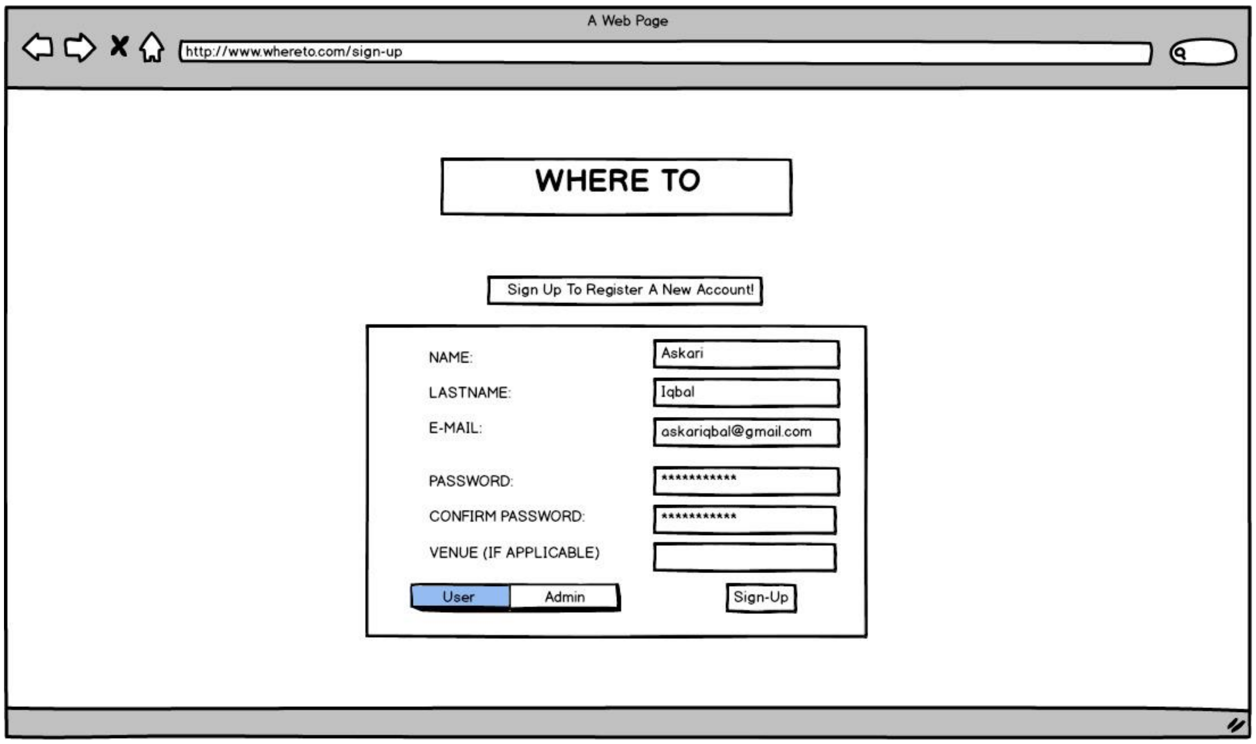


This is the log-in page where a user or admin can enter their existing credentials and access the application. If a user or admin is newly registering, then they may select the sign-up option at the bottom of the interface.

Corresponding SQL statements:

```
*************
Login-as-user
*************

Select (*)
From user
WHERE email = "<input>" AND password = "<input>"

*******************
Login-as-venue-admin
*******************

Select (*)
From venue_admin
WHERE email = "<input>" AND password = "<input>"
```

5.2 Sign-Up


A Web Page
http://www.whereto.com/sign-up
WHERE TO
Sign Up To Register A New Account!
[✓ User] Admin
Sign-Up

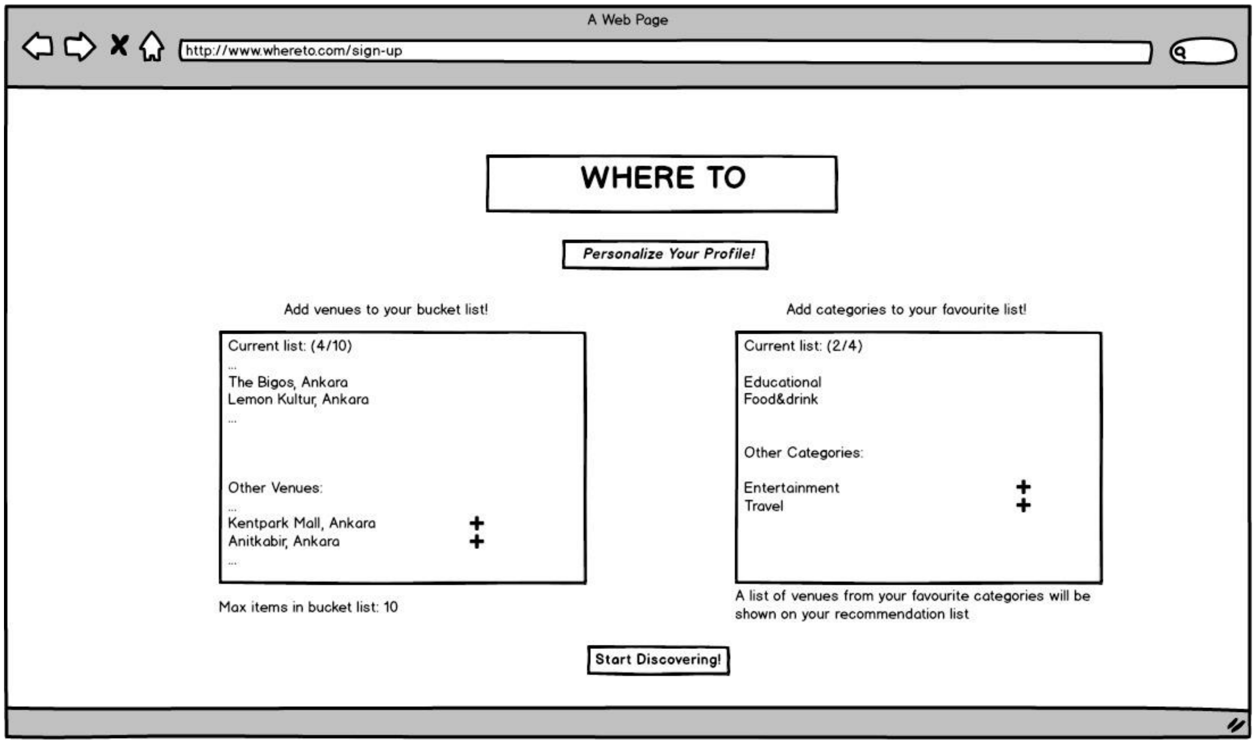
NAME:	Askari
LASTNAME:	Iqbal
E-MAIL:	askariqbal@gmail.com
PASSWORD:	***********
CONFIRM PASSWORD:	***********
VENUE (IF APPLICABLE)	

A user or admin can register a new account by providing a few details as shown below. If the admin option is selected then they must state the venue they are administrating into the data-base.

Corresponding SQL statements:

Insert into User(user_ID, name, password, email, food_drink, educational, entertainment, travel)
Values(“user_ID_generated”, “name + lastname”, “hashed password”, “email”, “boolean_user_response”, “boolean_user_response”, “boolean_user_response”, “boolean_user_response”)

5.3 Personalize Profile


A Web Page
http://www.whereto.com/sign-up
WHERE TO
Personalize Your Profile!
Add venues to your bucket list!
Add categories to your favourite list!
Start Discovering!

Current list: (4/10)
...
The Bigos, Ankara
Lemon Kultur, Ankara
...

Other Venues:
...
Kentpark Mall, Ankara +
Anitkabir, Ankara +
...

Max items in bucket list: 10

Current list: (2/4)

Educational
Food&drink

Other Categories:

Entertainment +
Travel +

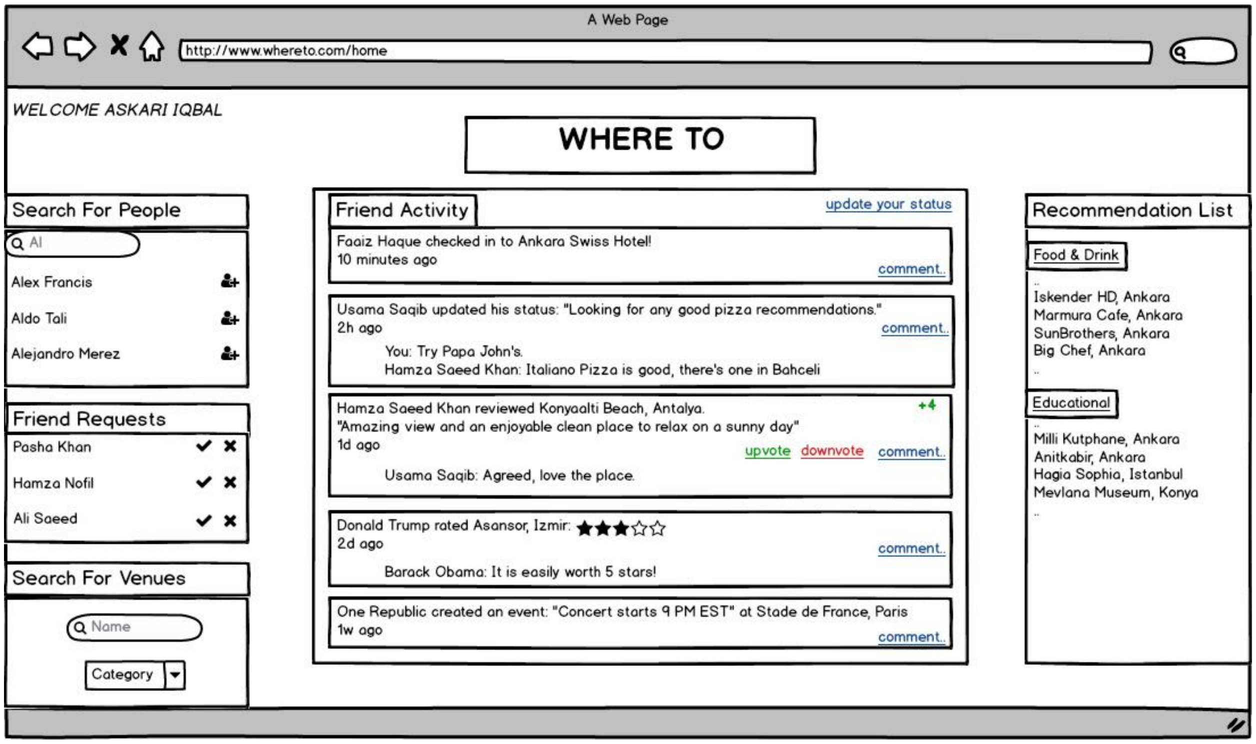
A list of venues from your favourite categories will be shown on your recommendation list

After registering an account, another sign-up step will be prompted to the user where they may add a list of venues they wish to visit to their bucket list. Additionally, they can mention their favorite categories in order to be recommended venues by the system that are relevant to their preferences.

Corresponding SQL statements:

```
********************
Personalized Profile
********************
Insert into user (bucket_list)
Values ( <input> )

Insert into user ( Food_and_drink, Educational, Entertainment, Travel)
Values ( <bool-1>, <bool-2>, <bool-3>, <bool-4>)
```

5.4 Home Page



A Web Page

http://www.whereto.com/home

WELCOME ASKARI IQBAL

WHERE TO

Search For People

Al

Alex Francis

Aldo Tali

Alejandro Merez

Friend Requests

Pasha Khan ✓ ✗

Hamza Nofil ✓ ✗

Ali Saeed ✓ ✗

Search For Venues

Name

Category

Friend Activity

update your status

Faaiz Haque checked in to Ankara Swiss Hotel!
10 minutes ago
comment..

Usama Saqib updated his status: "Looking for any good pizza recommendations."
2h ago
comment..
You: Try Papa John's.
Hamza Saeed Khan: Italiano Pizza is good, there's one in Bahceli

Hamza Saeed Khan reviewed Konyaalti Beach, Antalya.
"Amazing view and an enjoyable clean place to relax on a sunny day"
1d ago
+4
upvote downvote comment..
Usama Saqib: Agreed, love the place.

Donald Trump rated Asansor, Izmir: ★★★☆☆
2d ago
comment..
Barack Obama: It is easily worth 5 stars!

One Republic created an event: "Concert starts 9 PM EST" at Stade de France, Paris
1w ago
comment..

Recommendation List

Food & Drink

..
Iskender HD, Ankara
Marmura Cafe, Ankara
SunBrothers, Ankara
Big Chef, Ankara
..

Educational

..
Milli Kutphane, Ankara
Anitkabir, Ankara
Hagia Sophia, Istanbul
Mevlana Museum, Konya
..

The users are prompted to the home-page after successfully creating an account or logging in to an already existing account. As above we can see the user Askari Iqbal has logged in to his account. A lot of tasks are going on in the home page. The user can search for people, as shown above the characters 'Al' are typed into the search bar and a list of names starting with those two characters are shown. The user can then send them a friend request by clicking the friend icon besides their names. The user can also see the list of friend requests he has received and can choose to accept or reject them. If a user successfully befriends another user, he can see his activity on the home-page. Activity includes check-ins, statuses, reviews, rates, and event creations. Users can comment on all type of activities of their friends and further upvote and downvote a reviews. The user can also see their recommendation list on the right hand side which is based off their profiles that they personalized when signing up. Finally users can search for venues and filter them by a specified category.

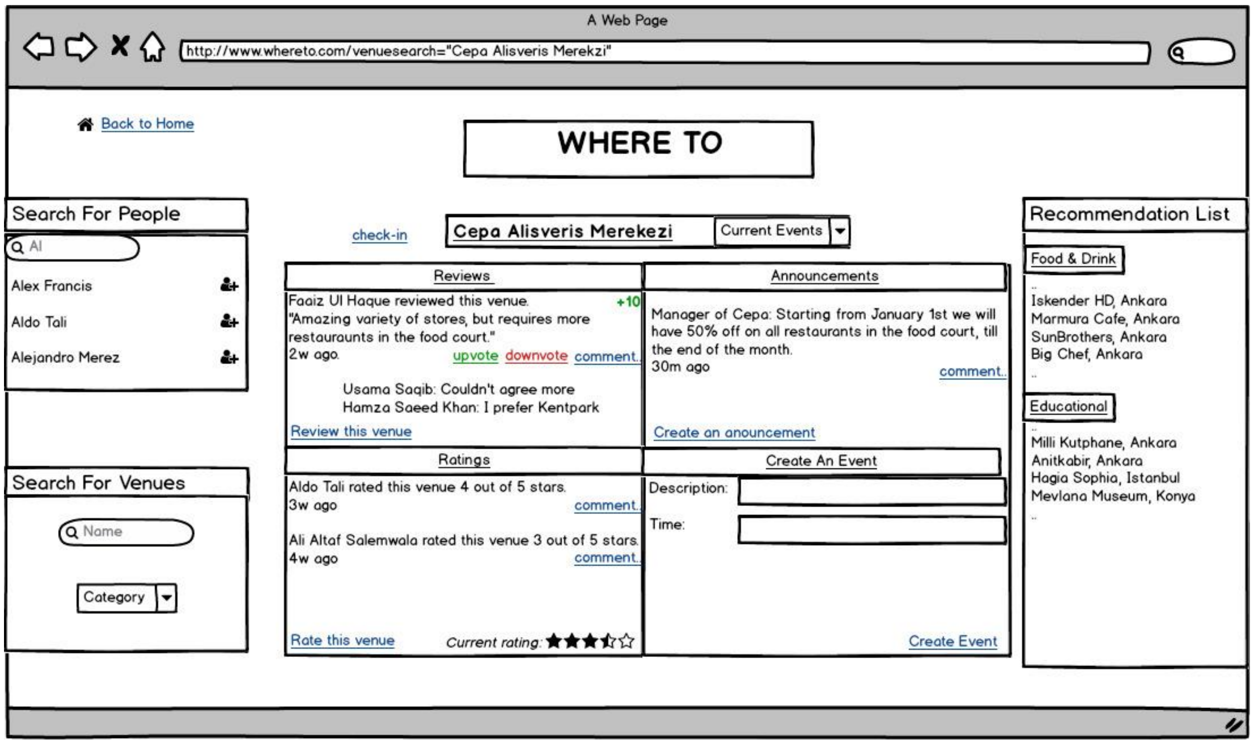
Corresponding SQL Statements:

```
*************
Search People
*************
Select (*)
```

```
From user
where name = "<input>"


**********
Add People
**********

Send Request
------------

Insert into befriend (friend_ID_1, friend_ID_2, friendship_status)
Values ("<input_ID>", "<input_ID>", "False")

Display Pending Requests
------------------------

Select (*)
From befriend
where ( (friend_ID_1 = "ID_of_user" OR friend_ID_2 = "ID_of_user")
           AND friendship_status = "False)

Accept Requests
---------------

Update befriend
SET friend_status = "True"
WHERE ( friend_ID_1 = "ID_of_request" AND friend_ID_2 =
"ID_of_recepient")
         OR ( friend_ID_1 = "ID_of_recepient" AND friend_ID_2 =
"ID_of_request")

Unfriend or Reject
------------------

Delete From befriend
WHERE ( friend_ID_1 = "ID_of_user" AND friend_ID_2 = "ID_of_block")
       OR (friend_ID_1 = "ID_of_block" AND friend_ID_2 = "ID_of_user")

Display Friends
---------------

Select (*)
FROM befriend
WHERE ( (friend_ID_1 = "ID_of_user" OR friend_ID_2 = "ID_of_user")
        AND friendship_status = "True)"

****************
Search for Venue
****************
Select (*)
From Venues
```

```
Where name = "<input_name>"




Show checkin of user
--------------------
Select (*)
FROM checkin
WHERE user-id = "ID"

******
Status
******

Delete-Status
-------------

Delete FROM Comment
WHERE status_ID = "status_ID"

Delete From Status
WHERE status_ID = "status_ID"

Display-Status
--------------
Select (*)
FROM status
where status_ID = (Select status_ID
                   FROM write_status
                   WHERE user_ID = "user_ID" )

Edit-Status
-----------
Update status
SET status_content = "new_status"
WHERE status_ID = "status_ID"

Comment-On-Status
-----------------
Insert into status-comment(status_ID, user_ID, content, time)
Values ("status_ID", "user_ID", "content", "time_stamp")

Display-Comments-on-Status
--------------------------
Select content, time
FROM status-comment
WHERE status_ID = "status_ID"
```

5.5 Venue



Faaiz Ul Haque reviewed this venue. +10
"Amazing variety of stores, but requires more restauraunts in the food court."
2w ago. upvote downvote comment..

Usama Saqib: Couldn't agree more
Hamza Saeed Khan: I prefer Kentpark

Review this venue

Manager of Cepa: Starting from January 1st we will have 50% off on all restaurants in the food court, till the end of the month.
30m ago comment..

Create an anouncement

Aldo Tali rated this venue 4 out of 5 stars.
3w ago comment..

Ali Altaf Salemwala rated this venue 3 out of 5 stars.
4w ago comment..

Rate this venue Current rating: ★★★⯪☆

Iskender HD, Ankara
Marmura Cafe, Ankara
SunBrothers, Ankara
Big Chef, Ankara

Milli Kutphane, Ankara
Anitkabir, Ankara
Hagia Sophia, Istanbul
Mevlana Museum, Konya

This page is prompted after a user uses the search for venue option and selects a venue. Users can also select a venue page from their friend's reviews or check-ins. Users can leave reviews on check-ins which receive comments and upvotes and downvotes. Users can also rate venues out of 5 stars. Users can also create events at the venue by specifying a time and a description of the event. Users can check-in to the venue as well. The admins can perform the same tasks as the users and in addition they can create announcements on the page.

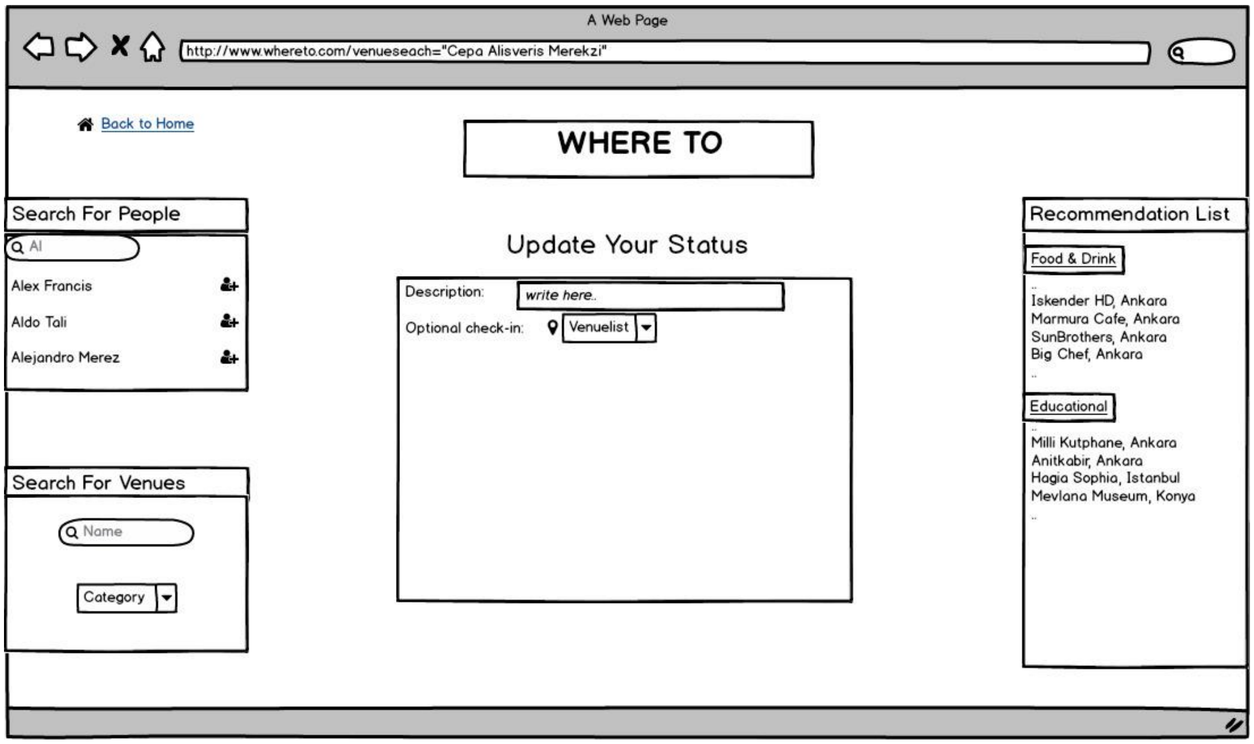
Corresponding SQL statements:

```
Display-rating
--------------

select average_rating
from ( Select avg(rating) as average_rating
       From rates
       group by venue_ID)
WHERE rate.venue_ID = "venue_ID

Write-Review
------------
Insert into Reviews( review_ID, content)
```

```
Values( "ID", "content")

Insert into write_review( writers_ID, review_ID)
Values( "user_ID", "review_ID")

Insert into has_review( venue_ID, review_ID)
Values("venue_ID", "review_ID")

Edit-Review
-----------
Update Reviews
SET content = "new_content"
WHERE review_ID = "ID"

Display-all-reviews-for-venue
-----------------------------

Select (*)
FROM Reviews
WHERE review_ID = ( Select review_ID
                   FROM has_review
                   WHERE venue_ID = "venue_ID" )
Display-review-votes
--------------------

//total upvotes
Select total_up_vote
From (Select count(vote)
      From Votes
       WHERE Vote = 1)
WHERE review_ID = "review_ID"

//total down votes
Select total_down_votes
FROM (Select count(vote)
      From Votes
      WHERE vote = 0)
WHERE review_ID = "review_ID"

Select (*)
FROM Votes
WHERE review_id = "ID"

Display-review-comments
-----------------------
Select content, time
From review_comment
WHERE review_ID = "ID"

Delete-Review
-------------
```

```
Delete from Votes
WHERE review-ID = "ID"

Delete From Comments
WHERE review-ID = "ID"

Delete From Reviews
WHERE review-ID = "ID"


Comment-on-review
-----------------
Insert into review_comment( review_ID, user_ID, content, time)
Values ( "review_ID", "user_ID", "content", time_stamp)


*****
Votes
*****

Display-votes-for-review
------------------------
Select upvote downvote
FROM votes
WHERE review_ID = "review_ID"

Check-if-user-has-voted-on-review
---------------------------------

Select vote
FROM votes
WHERE review_ID = "review_ID" AND user_id = "user_ID"

*****
RATES
*****

Insert into Rates( user_ID, venue_ID, rating)
Values( "user_ID", "venue_ID", "rating")

Display-all-reviews-of-user
---------------------------

Select (*)
FROM Reviews
WHERE review_ID = ( Select review_ID
                    From writes_review
                          WHERE writers_ID = "user_ID")
```

5.6 Status/Check-Ins



Users can write a status message and also check-in to a venue if they like. Users are prompted to this menu by either updating a status from the home-page or clicking on check-in from the venue page on a specified venue.

Corresponding SQL Statements:

```
Write-Status
------------
Insert into Status ( status_ID, status_content, time_stamp)
Values ("status_ID", "status_content", "time_stamp")

Insert into write-status ( user_ID, status_ID)
Values( "user_ID", "status_ID")

********
Check-in
********

Add Checkin
-----------
Insert into checkin( user-ID, venue-ID, time)
Values ("<input_user_ID>", "<input_venue_ID>", "time_stamp")
```

7. Advanced Database Components

a) Reports

To display the most liked Venues to a non user to browse. This allows for them to observe the most famous venues.

```
Select venue.name

from ( select venue.name , count (votes) as total_votes
        from rates, venue
        where rates.venue_ID = venue.venue_Id
        group by venue_id
      )

Order by total_votes DESC
```

To view the news feed of a user, an algorithm is used to find the newest status from their friends.

```
Select status_ID , status.time

from (  select status_ID , status.time
         from write-status
         where write-status.writer_ID = ( select user.user_ID
                                           from befriend
                                           where ((befriend.friend_ID_1 = current_User) AND NOT
(befriend.friend_ID_2 = current_User)) OR
((befriend.friend_ID_2 = current_User) AND NOT (befriend.friend_ID_1 = current_User))
                                          )
      )

group by status.time
```

Order by status.time DESC

Gets the total number of up votes of all the users for a review.

select count(vote) as upvotes

from votes

where review_id = current_review AND vote = 1

Gets the total number of down votes of all the users for a review.

select count(vote) as downvotes

from votes

where review_id = current_review AND vote = -1

Gets the average review of a venue by calculating all reviews of the users for a venue.

Select avg(rating)

from rates

where venue_ID = current_Venue

b) Views

This view is used to hide the user information from friend users.

Create View friends As

Select friend.user_ID

from befriend as friend

where
(friend.friend_ID_1 = currentUser AND NOT friend.friend_ID_2 = currentUser) OR
(friend.friend_ID_2 = currentUser AND NOT friend.friend_ID_1 = currentUser)

c) Triggers

- Upon creation of an event by a user, an invite is sent to all of the user's friends.
- Before voting on a review or rating on a view , it is checked if user has performed such an action before or not.
- Before writing a review, we check if the user has visited that place before or not.

d) Constraints

- Users need to sign in to change their password.
- Users need to sign in to comment and write a review.
- Users need to sign in to be able to vote for a review or rate a venue.
- Users need to have to check-in to be able to review or rate a venue.
- Venue needs to be owned by a venue admin.

8. Implementation Plan

For the DBMS system of our project we plan on using MySQL as it supports all modern features such as views, triggers, constraints, etc. As for the development technologies we are most likely going to use JavaScript and PHP for mainly interface and communication purposes. We are also considering using python or Java EE as they are native to us and contain libraries to make it easier for us to program a web application.