Bill Lynch
Erik Kamp
Jenny Shi

Project Step One

Description

Our application is a social networking site based on a common interest--music. Each user of the site can connect with friends and other users, and can make lists of songs, albums, and artists that they like. The lists can be sorted in a variety of ways--by title, album, artists, genre, and year. Each user can also interact with users by messaging them, and by searching users based on their hobbies outside of music.

System Requirements

Generic Functionalities

- List all of the users
  - input: *none*
  - input tables: users
  - output: list of users
- Display user profile (user information)
  - input: userid (or username)
  - input tables: user, attended, employment, interested_in, in_relationship_with, likesSong, likesAlbum, likesArtist
  - output: profile of each user
- Group users based on certain information (e.g. school)
  - input: school
  - input tables: user, attended
  - output: list of users attending the specified school
- Display user's message box (i.e. all of his/her messages)
  - input: userid (or username)
  - input tables: message
  - output: list of specified user's messages
- List user's friends
  - input: userid (or username)
  - input tables: user, friend
  - output: list of specified user's friends
- mutual friends (i.e. intersect two people's friends lists)
  - input: userid1, userid2 (or their usernames)
  - input tables: user, friend
  - output: list of common friends
- List user's pending requests
  - input: user
  - input tables: user, pending_friend
  - output: list of user's pending request
- List music (i.e. songs/albums/artists) a user likes
  - input: userid (or username)
  - input tables: user, likesSong (or likesArtist)
  - output: list of songs/artists the user likes
- List all people who like a certain song/album/artist

input: song (or album, artist)
input tables: user, likesSong (or likesAlbum, likesArtist)
output: list of all users who likes a certain song/album/artist

List songs by their attributes (album, artist, genre, etc.)
input: attribute (e.g. songs by a certain artist)
input tables: library
output: songs matching the attributes

List all of the songs/albums/artists/genres
input: *none*
input tables: library
output: list of songs (or album, artists, genre)

List of the most popular songs/albums/artists
input: *none*
input tables: song (or artist, album), likesSong (or likesAlbum, likesArtist)
output: list of songs (or album, artists) with the most likes

List user(s) with the largest song list (i.e. most number of liked songs)
input: *none*
input tables: user, song, likesSong; grouped by uid
output: user(s) with the max of number of songs liked

List all of the users under 30 employed at company A
input: age, company
input tables: user, company, employment
output: list of users

List the user's friends having at least one school in common with the user
input: user
input tables: user, school, attended, friend
output: list of friends

Output the average age of the users for each activity
input: *none*
input tables: user, activity, interested_in; grouped by actid
output: list of activities with their respective avg age

List the users in a relationship with another user from the same school
input: *none*
input tables: user, in_relationship_with, school, attended
output: list of users

List all single users that like a certain genre
input : genre
input tables : user, in_relationship_with, song
output: list of users

List all users in a certain town/state
input : state
input tables : user
output : list of users

List the top 10 most liked songs
input : none
input tables : user, likesong, song group by song
output : list of top songs

Relational Data Model

```
CREATE TABLE user(
        uid integer NOT NULL,
        photoid char(40) ,
        gender char(8) NOT NULL,
        address char(80),
        username char(32) NOT NULL,
        firstname char(32),
        lastname char(32),
        email char(32) NOT NULL,
        birthdate Date NOT NULL,
        PRIMARY KEY (userid),
        UNIQUE username
)

CREATE TABLE message(
        mid integer NOT NULL,
        ownerid integer NOT NULL,
        senderid integer NOT NULL,
        time datetime NOT NULL,
        content char(500),
        subject char(100) NOT NULL,

        PRIMARY KEY (mid),
        FOREIGN KEY ownerid REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY senderid REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE
)

CREATE TABLE friend(
        user1 integer NOT NULL,
        user2 integer NOT NULL,
        PRIMARY KEY (user1,user2),

        FOREIGN KEY user1 REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY user2 REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE
)

CREATE TABLE pending_friend (
        requesting integer NOT NULL,
```

```
        requested integer NOT NULL,

        PRIMARY KEY (requesting, requested),
        FOREIGN KEY requesting REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY requested REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE
)

CREATE TABLE in_relationship_with(
        user1 integer NOT NULL,
        user2 integer NOT NULL,
        status char(20),

        PRIMARY KEY (user1, user2),
        FOREIGN KEY user1 REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY user2 REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE
)

CREATE TABLE activity(
        aname char(30) NOT NULL,
        aid integer NOT NULL,

        PRIMARY KEY (aid)
)

CREATE TABLE interested_in (
        actid integer NOT NULL,
        uid integer NOT NULL,

        PRIMARY KEY (uid, actid),
        FOREIGN KEY uid REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY actid REFERENCES activity
                ON DELETE NO ACTION
                ON UPDATE CASCADE
)

CREATE TABLE school(
        sid integer NOT NULL,
        sname char(40) NOT NULL,
```

```
        address char(80),

        PRIMARY KEY (sid)
)

CREATE TABLE attended(
        uid integer NOT NULL,
        sid integer NOT NULL,
        start DATE,
        end DATE,
        degree char(40) NOT NULL,

        PRIMARY KEY (uid, sid, degree),
        FOREIGN KEY uid REFERENCES user,
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY sid REFERENCES school
                ON DELETE NO ACTION
                ON UPDATE CASCADE
)

CREATE TABLE company (
        cid integer NOT NULL,
        employer_name char(30) NOT NULL,
        address char(256),

        PRIMARY KEY (sid),
)

CREATE TABLE employment(
        job_title char(25),
        salary integer,
        uid integer NOT NULL,
        cid integer NOT NULL,

        PRIMARY KEY (uid, cid),
        FOREIGN KEY cid REFERENCES company
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY uid REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE
)

CREATE TABLE song (
        sid integer NOT NULL,
        sname char(30) NOT NULL,
        genre char(20),
```

```
        length integer,

        PRIMARY KEY (sid),
)

CREATE TABLE album(
        albumid char(30) NOT NULL,
        aname char(30) NOT NULL,
        year integer,

        PRIMARY KEY (albumid),
)

CREATE TABLE artist(
        aid integer NOT NULL,
        name char(30) NOT NULL,
        purl char(30),
        bio char(256),

        PRIMARY KEY (aid),
)

CREATE TABLE likesSong (
        uid integer NOT NULL,
        sid integer NOT NULL,

        PRIMARY KEY (userid,sid),
        FOREIGN KEY uid REFERENCES user,
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY sid REFERENCES song,
                ON DELETE CASCADE
                ON UPDATE CASCADE
)

CREATE TABLE likesAlbum (
        uid integer NOT NULL,
        albumid integer NOT NULL,

        PRIMARY KEY (userid,albumid),
        FOREIGN KEY uid REFERENCES user,
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY albumid REFERENCES album,
                ON DELETE CASCADE
                ON UPDATE CASCADE
)
```

```
CREATE likesArtist (
        uid integer NOT NULL,
        aid integer NOT NULL,

        PRIMARY KEY (userid,aid),
        FOREIGN KEY uid REFERENCES user
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY aid REFERENCES artist
                ON DELETE CASCADE
                ON UPDATE CASCADE
)

CREATE TABLE library (
        songid integer NOT NULL,
        albumid integer NOT NULL,
        artistid integer NOT NULL,

        PRIMARY KEY (songid, albumid, artistid),
        FOREIGN KEY songid REFERENCES song
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY albumid REFERENCES album
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY artist REFERENCES artist
                ON DELETE CASCADE
                ON UPDATE CASCADE,
)
```