Team Green -
Ohio State Parks Portal
Project Report

Adam
Bhavana
Gregory
Robert
Shruti

Introduction

BeFit Incorporated is a new state-funded company that is involved in getting people more involved and active in the community. The main goal of BeFit is to increase the fitness and overall health in our currently obese society. Given the amount of resources available to the public that allow for physical activities in a safe and desirable location, BeFit believes with a better knowledge database, the public will utilize the facilities at a much more frequent rate. More than 60% of adults do not get the suggested regular amount of physical activity and a staggering 50% of young adults, between ages 12 and 21, are not vigorously active on a regular basis. BeFit would like to increase the average amount of physical activity to 7 hours a week to achieve a decrease in early death rate by 40%.

To help achieve these goals, BeFit has hired Team Green to develop an application that allows the public access to data on state parks. Team Green has chosen state parks in Ohio to create a test application to judge the overall effectiveness and interaction of people before implementing a large scale project spanning the United States and including all state and national parks. With the test implementation, the service and data will be available for free to the clients.

1.3 Your team

Robert Zvolensky, Post-Bac Computer Science

- Extensive experience with XHTML, CSS, Photoshop / Graphic Design
- Skills also include:
    - PHP / mySQL
    - ASP.NET C# / mySQL
    - SQL table design / creation
    - Java Development

Adam Crosby, Junior Computer Science

-Experience with XHTML, Photoshop, Illustrator and C++.

Shruti Vangari, Masters in Computer Science

- C#,ASP,ADO,Adv .NET, MS SQL, Oracle SQL Developer, Java, HTML, RDBMS

Gregory A. Murphy, Jr., Junior Computer Science

-Experience with XHTML, C++, Java

Bhavana Kolli, Masters in Computer Science

- Experience with C, Java,HTML and Antrl

1. What are the primary functions of the application?

-The organization and storage of Ohio's state parks, trails and activities.

2. How many total end-users will this application support? What is the maximum number of concurrent users?

- This application will have a total amount of end users that reflects the state of Ohio's population plus the annual amount of tourism Ohio receives per year.

3. Where are the end-users located?

-Primarily Ohio, with the application available across the world.

4. How will the end-users access the application?

-Users can access via web application

5. What is the required availability of the application for end-users?

Application will be available 24/7

6. What is the acceptable downtime should disaster recovery be initiated?

4 hour recovery time objective (RTO) and a 2 hour recovery point objective (RPO)

Goal is to have hot standby at immediate instance

Recovery of main site is 4 hours

7. What are the response time requirements?

Current system response varies by API. Average response is between .03 - .20 with some transactions exceeding this number. Also dependant on user internet connection speed and bandwidth allocation.

8. Which platforms, operating systems and DBMSs can support the application?

-Linux platform with PHP implementation and DBMS mySQL.

9. Do you anticipate that there will be an initial load of data into the application from any source? If so, identify and describe. Also, will any data be redundantly stored from its original entry point?

Yes, initial data will be supplied by Ohio's state park website. The database will be approximately 150gig-500gig.

10. What data interfaces need to exist between data maintained by this application and data maintained by other applications? Will any application data be accessed, integrated with, or migrated to any other application?

The data will not need to be accessed by any other application having a full integration of web based elements with PHP and mySQL.

11. How long, and for what reasons, does the application data need to be retained? Can it be purged after that time or does it need to be archived? If archived, how will the data be retrieved when needed?

Data needs to be maintained indefinitely for the application to be running. Data archiving is not needed for the application.

12. Will sensitive data (e.g. credit card number, social security number) be stored? How will it be secured?

-No sensitive data will be stored.

13. What is the anticipated production date? What is the expected lifetime of the application?

-This exercise is for discovery only.

14. What is the expected volume of transactions by the end- users?

- Transaction volume will be determined by popularity of application of users in Ohio.

15. What is the expected growth of the application data?

- Over the next 5 years there is a projected growth of 0.1%.

16.What is the critical nature of the application data?

- This data is elements from the Ohio state parks and recreation website.

17. Are there any online (on-demand) reports? What are their expected volume and frequency?

- No, not at present for the re-platforming project.

18. Are there any batch (scheduled) reports? What are their expected volume and frequency?

- Yes, to get information on the amount of end-users using the application and feedback.

Possible Tables and Elements

asterisk(*) and underlined indicate keys

Parks(parkID, parkName, description, type, address, city, state, zip, phone, contactEmail, hours)

Park ID *
Park Name *
type - National Park/State Park
Description of Park
Address
Contact Information (phone, email)
Hours of Operation

Funding(fundingID, parkID, name, description, address, city, state, zip, phone, email, website)

Funding ID *
Funding Organization Name *
Park organization is funding
Contact Information

Directions(directionID, parkID, directionsText, GPScoords, linkToMap)

Direction ID *
Directions Text
GPS coordinates
URL to a map? (google maps)

Lodging(lodgeID, lodgeName, description, address, city, state, zip, phone, email, website, parkID)

LodgeID *
Lodge Name *
Lodge Description
Contact Information
Park that is located nearby

PetOptions(animalID, text)

animalID id of entry in db
text text description (i.e. Only Dogs, No Dogs, Only Horses, No Horses, Dogs and Horses, etc)

Trails(TRAILid, trailName, Description, length, difficulty, animalID, bikes, trailheadGPS, noticeID, parkID, numOfRatings, sumOfRatings, avgOfRatings)

Trail ID *
Trail Name *
Trail Description
Trail Length
Trail Difficulty
Bikes (bool)
animalID
Park Where Trail is Located
Any notices posted for this trail
GPS Coordinates

number of ratings
sum of total ratings
average ratings

AttractionType(typeID, name)

type ID *

attraction type * (hotel, campground, canoeing,etc)

Attractions(attractionID, name, description, address, city, state, zip, phone, email, website, photoFilename, parkID)

attractionID *
attraction name *
description
contactinformation
park located near by
maybe a photo of the attraction?

Notices(noticeID, noticeText, TRAILid)

noticeID *
notifications posted at the trail head
trail the notification is posted on.

UserRating(parkID, parkName, numOfRatings, sumOfRatings, avgOfRatings, bestTimeofVisit, thingsToCarry, reviews)

parkID *
parkName *
numOfRatings
sumOfRatings
avgOfRatings
bestTimeofVisit
thingsToCarry

reviews

CostnDeals(parkID, childCost, adultCost, petCost, groupPackage, otherDeals)

parkID *
childCost - Cost of the ticket per child
adultCost - Cost of the ticket per adult
petCost - Cost for pets if any
groupPacakge - Cost if there are group packages available
otherDeals - If there are deals and coupons for a particular time of the year etc.

Activities(parkID, walking, biking, swimming, fishing, hunting, iceFishing, camping, BBQ, sailing, kayaking, canoeing,)

parkID
walking(bool)
biking(bool)
swimming(bool)
fishing(bool)
hunting(bool)
iceFishing(bool)
camping(bool)
BBQ(bool)
sailing(bool)
kayaking(bool)
canoeing(bool)

SpecialEvents(parkID, eventID, eventName, eventType, eventDescription, eventDuration, datesOfOperation, eventLocation, eventCost, contactDetails)

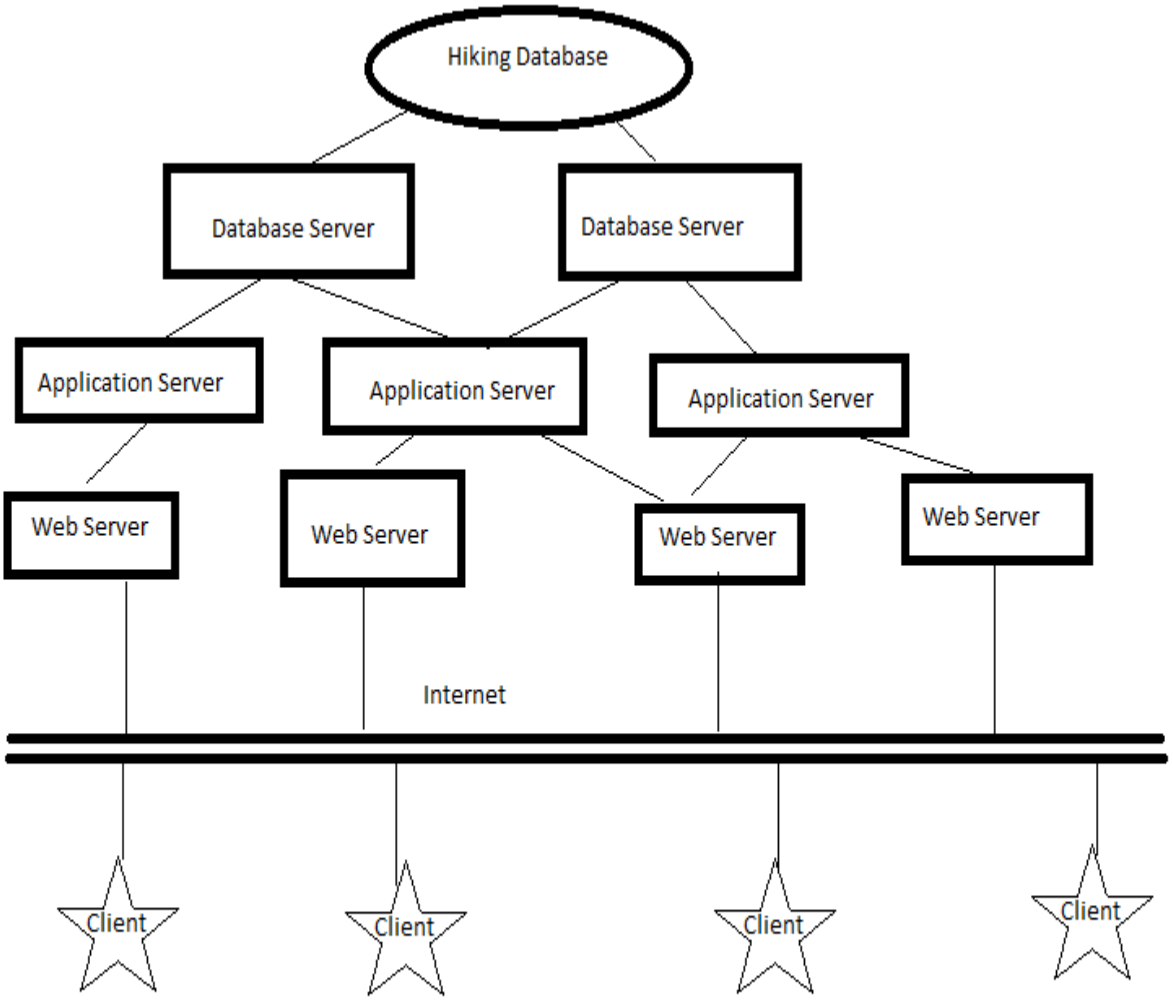
if there are special events being held at the park.
parkID
eventID
eventName
eventType
eventDescription

eventDuration
datesOfOperation
eventLocation
eventCost
contactDetails

additionalAmenities(parkID, wheelchairAvailability, wheelchairLocationAvailability, locationOfFirstAid, emergencyContactNumber)

parkID
wheelchairAvailaibility
wheelchairLocationAvailability
locationOfFirstAid
emergencyContactNumber

Analysis of Requirements



Relational Algebra Example

Client is looking for park with a 5km trail

Trails

Trail ID	Trail Name	Description	Length	Difficulty	AnimallD	Bikes	Trailhead GPS	noticeID	Park ID	numberofRatings	sumOfRatings	averageOfRatings
123654	BoatHead	Outdoorsy	5km	medium	1	0	NULL	NULL	5542	50	9	8
90210	Bower	Scenic	5km	easy	1	1	NULL	NULL	1235	89	5	4

Sigma(Length) ≥ 5(Trails)

The relational algebra is important to allow the client to easily use the search function to narrow results to available and pertinent information requested.

Phase 2: Database Descriptions

2.1. Objective

BeFit Incorporated is a new state-funded company that is involved in getting people more involved and active in the community. The main goal of BeFit is to increase the fitness and overall health in our currently obese society. Given the amount of resources available to the public that allow for physical activities in a safe and desirable location, BeFit believes with a better knowledge database, the public will utilize the facilities at a much more frequent rate.

The current database has around 15 tables which would allow a user to review details on a particular park, learn about possible trails available in the park and prepare himself/herself accordingly. The user will be able to search for specific criteria to narrow down park choices to tailor the users interests. The user

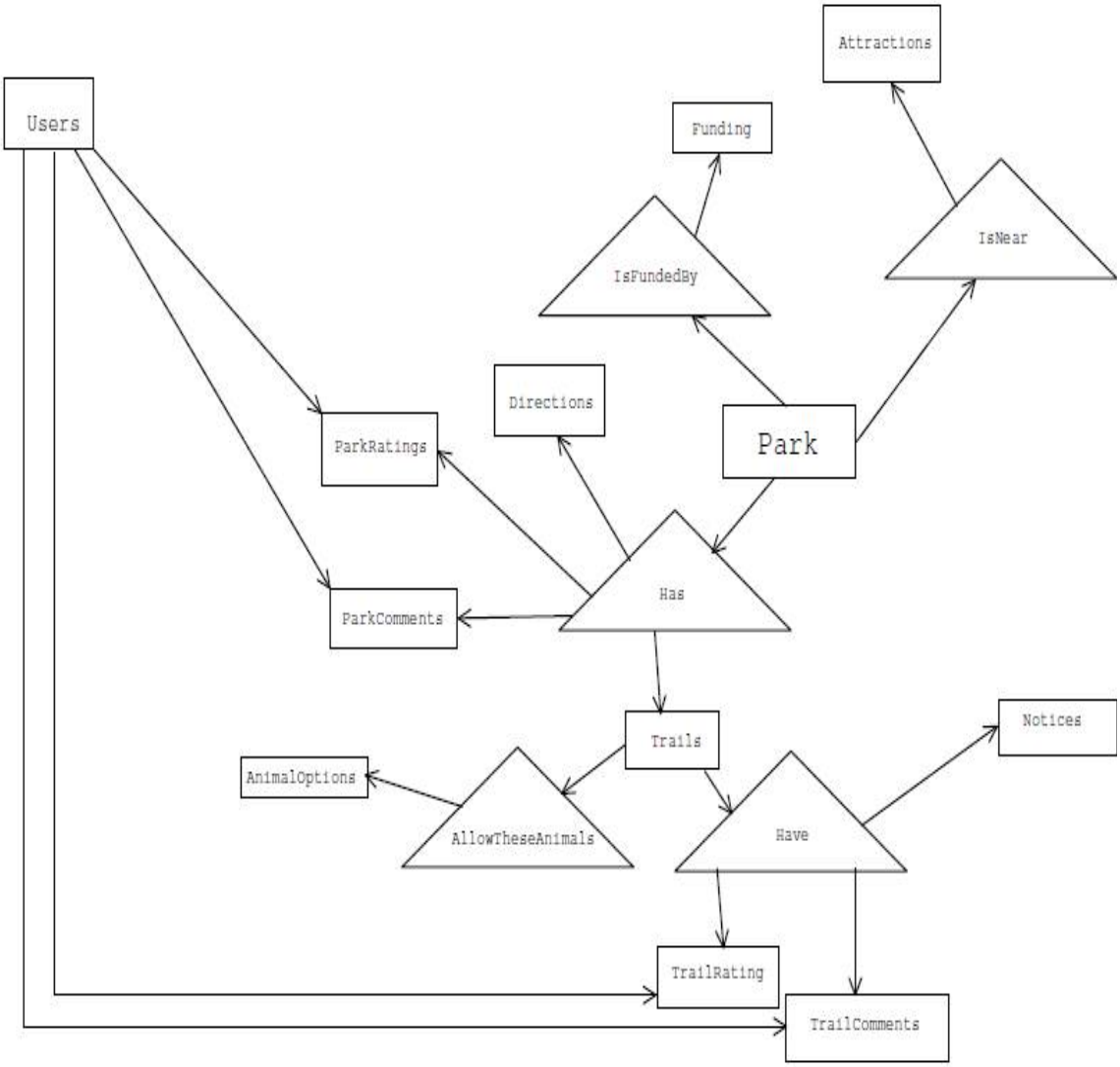
can also learn about possible pet options, directions to the park, distance and the types of attractions around the park. Once the user visits the park, he can finally rate the trail/park for other viewers to review.

2.2. DBMS Environment

A Relational database management system has been selected to store the data in the form of tables. The front end would be written using PHP and the website can pull data from the backend. The software used for the backend is MS SQL - which interfaces easily with PHP. Hence, we have chosen the same - for faster retrieval and easy storage.

2.3. Database Design and Data Models

Part 1: ER Diagram



Part 2 : Decomposition into BCNF

User(<u>UserID</u>, UserName, Pword, FirstName, LastName)
{UserID}+ = {UserID UserName, Pword, FirstName, LastName}
UserId→UserName
UserID→Pword
UserId→FirstName

UserID→FirstName
UserID→UserName Pword FirstName LastName
{UserID UserName, Pword, FirstName, LastName}

Park(ParkID, ParkName, ParkAddress, ParkCity, ParkState, ParkZip)
{ParkID}+ = {ParkID ParkName, ParkAdress, ParkCity, ParkState, ParkZip}
ParkId→ParkName
ParkID→ParkAdress
ParkId→ParkCity
ParkID→ParkState
ParkID→ParkZip
ParkID→ParkName ParkAdress ParkCity ParkState ParkZip
{ParkID ParkName, ParkAdress, ParkCity, ParkState, ParkZip}

RatingsForPark(RatingID, RatedParkID, NumOfRatings, TotalRatings, MaxPossibleRating)
{RatingID }+ = { RatingID RatedParkID NumOfRatings, TotalRatings, MaxPossibleRating}
RatingID → RatedParkID
RatingID → NumOfRatings
RatingID → TotalRatings
RatingID → MaxPossibleRating
RatingID → RatedParkID NumOfRatings TotalRatings MaxPossibleRating
{ RatingID RatedParkID NumOfRatings, TotalRatings, MaxPossibleRating}

CommentsForPark (CommentID, CommentForParkID, CommenterID, CommentText)
{CommentID}+ = {CommentID CommentForParkID CommenterID CommentText}
CommentID→ CommentForParkID
CommentID→ CommenterID
CommentID→ CommentText
CommentID→ CommentForParkID CommenterID
{CommentID CommentForParkID CommenterID CommentText}

DirectionsToPark(DirectionsID, DirectionsToParkID, DirectionFrom, DirectionText, GPScoords)
{DirectionsID}+ = {DirectionsID DirectionsToParkID DirectionsFrom DirectionText GPScoords}
DirectionsID → DirectionsToParkID
DirectionsID → DirectionsFrom

DirectionsID → DirectionText
DirectionsID → GPScoord
{DirectionsID DirectionsToParkID DirectionsFrom DirectionText GPScoords}

TrailDifficulty(diffID, difficultyWord, difficultyDescription)
{diffID}+ = {diffID difficultyWord difficultyDescription}
diffID → difficultyWord
diffID → difficultyDescription
diffID →difficultyWord difficultyDescription
{diffID difficultyWord difficultyDescription}

Trail(TrailID, TrailParkID, TrailLength, Difficulty, GPScoords)
{TrailID}+ = {TrailID TrailParkID TrailLength Difficulty GPScoords}
TrailID→TrailParkID
TrailID→TrailLength
TrailID→Difficulty
TrailID→GPScoords
TrailID→ TrailParkID TrailLength Difficulty GPScoords
{TrailID TrailParkID TrailLength Difficulty GPScoords}

RatingsForTrail(RatingID, RatedTrailID, NumOfRatings, TotalRatings, MaxPossibleRatings)
{RatingID }+ = { RatingID RatedTrailID NumOfRatings TotalRatings MaxPossibleRating}
RatingID → RatedTrailID
RatingID →NumOfRatings
RatingID →TotalRatings
RatingID →MaxPossibleRating
RatingID → TrailID NumOfRatings TotalRatings MaxPossibleRating
{ RatingID RatedTrailID NumOfRatings TotalRatings MaxPossibleRating}

CommentsForTrail(CommentID, CommentForParkID, CommenterID, CommentText)
{CommentID}+ = { CommentID, CommentForParkID, CommenterID, CommentText }
CommentID → CommentForParkID
CommentID → CommenterID
CommentID → CommentText
CommentID → CommentForParkID, CommenterID, CommentText
{ CommentID, CommentForParkID, CommenterID, CommentText }

TrailNotices(NoticeID, NoticeForTrailID, NoticeText, displayed)
{NoticeID}+ = { NoticeID NoticeForTrailID NoticeText displayed}
NoticeID →NoticeForTrailID
NoticeID →NoticeText
NoticeID →Displayed
NoticeID → NoticeForTrailID NoticeText displayed
{ NoticeID NoticeForTrailID NoticeText displayed}

PetsOnTrail(PetID, ForTrailID, type)
{PetID}+ = { PetID ForTrailID type}
PetID → ForTrailID
PetID → ForTrailID type
{ PetID ForTrailID type}

PetType(PetTypeID, type)
{PetTypeID }+ = { PetTypeID type}
PetTypeID →type
{ PetTypeID type}

AttractionsNearPark(AttractionID, ParkIDNear, AttName, AttAddress, AttCity, AttState, AttZipCode, AttDescription, AttWebsite, AttPhone, AttractionType, PetFriendly, GPScoords)
{AttractionID}+ = { AttractionID ParkIDNear AttName AttAddress AttCity AttState, AttZipCode AttDescription AttWebsite AttPhone AttractionType PetFriendly GPScoords }
AttractionID → ParkIDNear
AttractionID → AttName
AttractionID → AttAddress
AttractionID → AttCity
AttractionID → AttAttState
AttractionID → AttState
AttractionID→ AttZipCode
AttractionID→ AttDescription
AttractionID → AttWebsite
AttractionID → AttPhone
AttractionID → AttType
AttractionID → PetFriendly
AttractionID → GPScoords

AttractionID → ParkIDNear ParkIDNear AttName AttAddress AttCity AttState, AttZipCode AttDescription AttWebsite AttPhone AttractionType PetFriendly GPScoords
{ AttractionID ParkIDNear AttName AttAddress AttCity AttState, AttZipCode AttDescription AttWebsite AttPhone AttractionType PetFriendly GPScoords }

TypeOfAttraction(TypeID, type)
{TypeID}+ = {TypeID type}
TypeID→ type
{TypeID type}

FundingForPark(FundID, FundedParkID, FundName, FundAddress, FundCity, FundState, FundZipCode, FundDescription, FundWebSite, FundPhone)
{FundID}+ = { FundID FundedParkID FundName FundAddress FundCity FundState FundZipCode FundDescription FundWebSite FundPhone }
FundID→FundedParkID
FundID→FundName
FundID→FundAddress
FundID→FundCity
FundID→FundState
FundID→ZipCode
FundID→FundDescription
FundID→FundWebSite
FundID→FundPhone
FundID→FundName FundAddress FundCity FundState FundZipCode FundDescription FundWebSite FundPhone
{ FundID FundedParkID FundName FundAddress FundCity FundState FundZipCode FundDescription FundWebSite FundPhone }

Part 3: ER to Relational Schema

2.4. Physical Database Design

2.5. Constraints, Foreign Keys and Triggers-

DDL Statements:

--**********CREATE STATEMENTS**********

```
CREATE TABLE Users(
userID int PRIMARY KEY,
username varchar(30) NOT NULL,
Pword varchar(20) NOT NULL,
FirstName varchar(20) NOT NULL,
LastName varchar(20) NOT NULL)

CREATE TABLE Park(
ParkID INT PRIMARY KEY,
ParkName VARCHAR(100) NOT NULL,
ParkAddress VARCHAR(300) NOT NULL,
ParkCity VARCHAR(100) NOT NULL,
ParkState VARCHAR(100) NOT NULL,
ParkZipCode INT NOT NULL)

CREATE TABLE RatingsForPark(
RatingID INT Primary Key,
RatedParkID INT REFERENCES PARK(ParkID) ON DELETE CASCADE,
ReceivedRatings INT NOT NULL,
PossibleRatings INT NOT NULL)

CREATE TABLE CommentsForPark(
CommentID INT Primary Key,
CommentForParkID INT REFERENCES PARK(ParkID) ON DELETE
CASCADE,
CommenterID INT REFERENCES USERS(UserID) ON DELETE CASCADE,
CommentText VARCHAR(1000) NOT NULL)

CREATE TABLE DirectionsToPark(
DirectionsID INT Primary Key,
DirectionsToParkID INT REFERENCES PARK(ParkID) ON DELETE
CASCADE,
DirectionFrom VARCHAR(400), -- NORTH, SOUTH, EAST, WEST
DirectionText VARCHAR(8000),
GPScoords VARCHAR(100))
```

```
CREATE TABLE TrailDifficulty(
diffID INT Primary Key,
difficultyWord VARCHAR(100) NOT NULL,
difficultyDescription VARCHAR(100) NOT NULL)

CREATE TABLE Trail(
TrailID INT PRIMARY KEY,
TrailName VARCHAR(200),
TrailInParkID INT REFERENCES PARK(ParkID) ON DELETE CASCADE,
TrailLength INT NOT NULL,
Difficulty INT REFERENCES TrailDifficulty(diffID) ON DELETE SET NULL,
GPScoords VARCHAR(100))

CREATE TABLE RatingsForTrail(
RatingID INT Primary Key,
RatedTrailID INT REFERENCES Trail(TrailID) ON DELETE CASCADE,
ReceivedRatings INT NOT NULL,
PossibleRatings INT NOT NULL)

CREATE TABLE CommentsForTrail(
CommentID INT Primary Key,
CommentForParkID INT REFERENCES Trail(TrailID) ON DELETE
CASCADE,
CommenterID INT REFERENCES USERS(UserID) ON DELETE CASCADE,
CommentText VARCHAR(5000) NOT NULL)

--CREATE TABLE TrailNotices(
--NoticeID INT Primary Key,
--NoticeForTrailID INT REFERENCES Trail(TrailID) ON DELETE CASCADE,
--NoticeText VARCHAR NOT NULL,
--displayed char) -- show/hide option

CREATE TABLE PetType(
PetTypeID INT Primary Key,
type VARCHAR(100) NOT NULL) -- only dogs, both dogs and horses, etc

CREATE TABLE PetsOnTrail(
petID INT Primary Key,
ForTrailID INT REFERENCES Trail(TrailID) ON DELETE CASCADE,
```

```
type INT REFERENCES PetType(PetTypeID) ON DELETE CASCADE)

CREATE TABLE TypeOfAttraction( -- Lodging, Camping and Recreation
TypeID INT Primary Key,
typeof VARCHAR(255) NOT NULL)

--
--CREATE TABLE FundingForPark(
--FundID INT Primary Key,
--FundedParkID INT REFERENCES Park ON DELETE CASCADE,
--FundName VARCHAR NOT NULL,
--FundAddress VARCHAR NOT NULL,
--FundCity VARCHAR NOT NULL,
--FundState VARCHAR NOT NULL,
--FundZipCode INT NOT NULL,
--FundDescription VARCHAR NOT NULL,
--FundWebsite VARCHAR)

CREATE TABLE AttractionsInPark(
AttractionID INT Primary Key,
InParkID INT REFERENCES Park(ParkID) ON DELETE CASCADE,
AttractionType INT REFERENCES TypeOfAttraction(TypeID))

--**********INSERT STATEMENTS**********
INSERT INTO Park VALUES ('Hocking Hills State Park', '19852 State Route
664 S','Logan','Ohio',43138);
INSERT INTO Park VALUES ('Jackson Lake State Park', '35 Tommy Been
Road','Oak Hill','Ohio',45656);
INSERT INTO Park VALUES ('Cleveland Lakefront State Park', '8701
Lakeshore Blvd, NE','Cleveland','Ohio',44108);
INSERT INTO Park VALUES ('Portage Lakes State Park', '5031 Manchester
Road','Akron','Ohio',44319);
INSERT INTO Park VALUES ('Tinkers Creek State Park', '10303 Aurora
Hudson Rd.','Streetsboro','Ohio',44241);
INSERT INTO Park VALUES ('Lake Milton State Park', '16801 Mahoning
Avenue','Lake Milton','Ohio',44429);
INSERT INTO Park VALUES ('Buckeye Lake State Park', '2905 Liebs Island
Road','Millersport','Ohio',43046);
INSERT INTO Park VALUES ('Mosquito Lake State Park', '1439 State Route
305','Cortland','Ohio',44410);
```

```
INSERT INTO Park VALUES ('Beaver Creek State Park', '12021 Echo Dell Road','East Liverpool','Ohio',43920);
INSERT INTO Park VALUES ('Deer Creek State Park', '20635 State Park Road 20','Mt. Sterling','Ohio',43143);
INSERT INTO Park VALUES ('Alum Creek State Park', '3615 S. Old State Road','Delaware','Ohio',43015);
INSERT INTO Park VALUES ('Blue Rock State Park', '7924 Cutler Lake Road','Blue Rock','Ohio',43720);
INSERT INTO Park VALUES ('Buck Creek State Park', '1901 Buck Creek Lane','Springfield','Ohio',45502);
INSERT INTO Park VALUES ('Caesar Creek State Park', '8570 E State Route 73','Waynesville','Ohio',45068);
INSERT INTO Park VALUES ('Hueston Woods State Park', '6301 Park Office Road','College Corner','Ohio',45003);
INSERT INTO Park VALUES ('Lake Loramie State Park', '4401 Ft. Loramie Swanders Road','Minster','Ohio',45865);
INSERT INTO Park VALUES ('Mohican State Park', '3116 State Route 3','Loudonville','Ohio',44842);
INSERT INTO Park VALUES ('Mt Gilead State Park', '4119 State Route 95','Mt. Gilead','Ohio',43338);
INSERT INTO Park VALUES ('Sycamore State Park', '4675 N. Diamond Mill Road','Trotwood','Ohio',45426);
INSERT INTO Park VALUES ('Nelson-Kennedy Ledges State Park', 'State Route 282','Nelson Township','Ohio',44065);
INSERT INTO Park VALUES ('Wingfoot Lake State Park', '993 Goodyear Park Blvd','Mogadore','Ohio',44260);

INSERT INTO DirectionsToPark
VALUES(1300,100,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1301,101,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1302,102,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1303,103,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1304,104,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1305,105,'ToEnter','ProvideDirections','ProvideCoords');
```

```
INSERT INTO DirectionsToPark
VALUES(1306,106,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1307,107,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1308,108,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1309,109,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1310,110,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1311,111,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1312,112,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1313,113,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1314,114,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1315,115,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1316,116,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1317,117,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1318,118,'ToEnter','ProvideDirections','ProvideCoords');
INSERT INTO DirectionsToPark
VALUES(1319,119,'ToEnter','ProvideDirections','ProvideCoords');

--Taken from http://www.imba.com/resources/maps/trail-difficulty-ratings

INSERT INTO TrailDifficulty VALUES('White Circle','Easiest');
INSERT INTO TrailDifficulty VALUES('Green Circle','Easy');
INSERT INTO TrailDifficulty VALUES('Blue Square','More Difficult');
INSERT INTO TrailDifficulty VALUES('Black Diamond','Very Difficult');
INSERT INTO TrailDifficulty VALUES('Dbl. Black Diamond','Extremely
Difficult');

INSERT INTO Trail Values(10000,'Mountain Bike',110,2,90000,NULL)
INSERT INTO Trail Values(10001,'Mountain Bike',110,5,90002,NULL)
```

```
INSERT INTO Trail Values(10002,'Mountain Bike',110,7,90003,NULL)
INSERT INTO Trail Values(10003,'Park Office trail',110,1.5,90000,NULL)
--INSERT INTO Trail Values(10004,'Hollen Back',110,NULL,NULL,NULL)
INSERT INTO Trail Values(10005,'Multipurpose',110,7,90003,NULL)
INSERT INTO Trail Values(10006,'Bridle trails',110,38,90002,NULL)

INSERT INTO TypeOfAttraction VALUES('Lodging')
INSERT INTO TypeOfAttraction VALUES('Camping')
INSERT INTO TypeOfAttraction VALUES('Recreation')
INSERT INTO TypeOfAttraction VALUES('Food')

INSERT INTO Attractions VALUES('Longhorn Steakhouse', '111 Washington
Blvd.','Wooster','OH', 44691,NULL,'http://www.longhorn.com', '(330) 555-1234',
4)
INSERT INTO Attractions VALUES('Swensons', '24 Howe Ave','Akron','OH',
44245,NULL,'http://www.swensons.com','(330) 555-2345', 4)
INSERT INTO Attractions VALUES('Putt n Stuff', '53 Acre
Drive','Wooster','OH', 44691, NULL, NULL,'(330) 555-3456', 3)
INSERT INTO Attractions VALUES('Mrs Millers Cabin', '455 State Route
557','Charm','OH', 44623, NULL, NULL,'(330) 555-3456', 1)
INSERT INTO Attractions VALUES('Camp Toodik', '5601 State Route
39','Loudonville','OH', 44605, NULL, NULL,'(330) 555-3456', 2)
INSERT INTO Attractions VALUES ('Conkles Hollow','24858 Big Pine
Rd','Rockbridge','OH',43149,NULL, NULL,'(614) 265-6561',1)
INSERT INTO Attractions VALUES ('Olentangy Caverns','1779
Home Rd','Delaware','OH',43015,NULL,'http://
www.olentangyindiancaverns.com/','(740) 548-7917',2)
INSERT INTO Attractions VALUES ('Kyle Woods','6920 Tippecanoe
Rd','Canfeild','OH',44406,NULL,NULL,'(614) 265-6561',3)
INSERT INTO Attractions VALUES ('U.S. Air Force Museum','1100 Spaatz
Street','Dayton','oh',45431,NULL,'www.nationalmuseum.af.mil/','(937) 255-
3286',3)
INSERT INTO Attractions VALUES ('Kings Island amusement park','Virtual
Address','Mason','OH',45039,NULL,'http://www.visitkingsisland.com/',NULL,2)
INSERT INTO Attractions VALUES ('city of Mansfield','124 North Main
Street','Mansfields','OH',44902,NULL,'http://www.mansfieldtourism.com/','800-
642-8282',1)

INSERT INTO AttractionsNearPark VALUES(1, 2)
INSERT INTO AttractionsNearPark VALUES(4, 5)
```

```
INSERT INTO AttractionsNearPark VALUES(1, 6)
INSERT INTO AttractionsNearPark VALUES(1, 1)
INSERT INTO AttractionsNearPark VALUES(2, 1)
INSERT INTO AttractionsNearPark VALUES(3, 1)
INSERT INTO AttractionsNearPark VALUES(4, 1)
INSERT INTO AttractionsNearPark VALUES(5, 1)

INSERT INTO PetType VALUES('Only Dogs')
INSERT INTO PetType VALUES('Only Horses')
INSERT INTO PetType VALUES('No Pets')
INSERT INTO PetType VALUES('Dogs and Horses')

INSERT INTO PetsOnTrail VALUES(1,1);

INSERT INTO Trail Values('Mountain Bike',1,2,1,NULL)
INSERT INTO Trail Values('Mountain Bike',2,5,3,NULL)
INSERT INTO Trail Values('Mountain Bike',3,7,4,NULL)
INSERT INTO Trail Values('Park Office trail',4,1.5,1,NULL)
INSERT INTO Trail Values('Multipurpose',5,7,4,NULL)
INSERT INTO Trail Values('Bridle trails',6,38,2,NULL)
INSERT INTO Trail VALUES ('Orange Trail Loop',1,2,4,NULL)
INSERT INTO Trail VALUES ('Buckeye Trail',1,6,2,NULL)
INSERT INTO Trail VALUES ('Cantwell Cliffs',1,1,3,NULL)
INSERT INTO Trail VALUES ('Pond Run Trail',5,1.5,1,NULL)
INSERT INTO Trail VALUES ('Lake-view Trail',5,0.75,2,NULL)
INSERT INTO Trail VALUES ('Nature Trail',6,1.25,1,NULL)
INSERT INTO Trail VALUES ('Oak Hill Trail',9,1,2,NULL)
INSERT INTO Trail VALUES ('Dogwood Trail',9,2,3,NULL)
INSERT INTO Trail VALUES ('Lower Vondergreen Trail',9,3.25,3,NULL)
INSERT INTO Trail VALUES ('bridle trails',9,23,4,NULL)
INSERT INTO Trail VALUES ('Rolling Hills Trail',10,2.5,3,NULL)
INSERT INTO Trail VALUES ('Mountain Biking',10,1,1,NULL)
INSERT INTO Trail VALUES ('Bridle Trail',10,17,2,NULL)
INSERT INTO Trail VALUES ('Deer Trail',12,0.2,4,NULL)
INSERT INTO Trail VALUES ('bridle trails',12,26,3,NULL)
INSERT INTO Trail VALUES ('Ground Cedar Trail',12,0.45,2,NULL)
INSERT INTO Trail VALUES ('Buckhorn Trail',13,7.5,3,NULL)
INSERT INTO Trail VALUES ('Silidago Downs Trail',14,28,1,NULL)
INSERT INTO Trail VALUES ('Spillway Trail',14,4.3,3,NULL)
INSERT INTO Trail VALUES ('Bridle Trail',15,18,3,NULL)
```

```
INSERT INTO Trail VALUES ('West Shore Trail',15,6,2,NULL)
INSERT INTO Trail VALUES ('Little Turtle Trace',16,1,1,NULL)
INSERT INTO Trail VALUES ('Hemlock Gorge Trail',17,2,4,NULL)
INSERT INTO Trail VALUES ('Fern Ridge Trail',18,1,4,NULL)
INSERT INTO Trail VALUES ('Ghost Hedge Trail',19,2.97,2,NULL)
INSERT INTO Trail VALUES ('Red Trail',20,0.5,5,NULL)
```

2.6. Views and Indexes

Cursor Processing will be used for Views

- o Trails in a park
- o Attractions near a park
- o General list of Parks
- o Calculating Average Rating of Parks and Trails
- o Display all comments for a particular Park or Trail

Advantages of views:

1. Security - Helps us to hide the entire details of the table and only projects the necessary rows and columns.
2. Simplicity - Hides the complicated structure of the database and its schema and allows a user to view the requisite data.

To view comments about a particular trail - Example

```
Create VIEW CommentsForATrail AS
(select CommentText from CommentsForTrail where
CommentForParkID in
(Select TrailID from Trail where TrailName = 'Bridle trails'))

Select * from CommentsForATrail
```

Use of Indexes:

Helps retrieve data quickly and stores it in a less-complex way in the database.

2.7. Interface and Stored Procedures

Primary interface we will be using: Call-Level Interface (CLI)
- Integrated with PHP
- SQL Queries made when page(s) are (re-)loaded.
These queries can also call a stored procedure.

Stored Procedures (used by triggers):
- Creation of users (check to see if username is already in users DB)
- User Login Authentication
- Password Encryption?
- Updates (Not a part of initial release of project)
o User Account updates (passwords)
o User capability to edit/delete comments he/she made

2.8. Alpha test plans

1. Model Testing - Since model testing is done by the most 'micro' scale of testing; to test particular table / class or any code., we intend to use Select statements to test the functionality of a table. Yet to work on a finale module test.
2. Integration testing - With the help of a log file and Microsoft SQL Server 2008/2010, we intend to write integration testing scripts to test the database. Also since we do not have access to each other database, we intend to take the different modules that each of us writes and integrate them into one and then test it.
3. Security testing - There are two ways to connect to a SQL Server - Windows authentication and SQL Server authentication. SQL Server authentication already provides

security and prevents outside users from accessing the database. MS SQL Server also has various features which would help us achieve this.

4. Backup and recovery testing - We intend to maintain a back up on a external hard drive. Apart from this, we also wish to replicate the data from the local CS machine to another local server - personal laptop perhaps.
5. End-to-end testing - Front-end and back-end integration testing would help us achieve this. Also, we would be using transaction processing - COMMIT, ROLLBACK transaction statements to recover any loss that might occur due to a breakdown.

Phase 3: Database Creation

Creation requirements and minimums required with compatibility information, if any.

3.1. Creation Instructions

```
--select * from sys.tables order by create_date desc

--Drop in the following order
Drop table DirectionsToPark
Drop table CommentsForPark
Drop table RatingsForPark
Drop table AttractionsInPark
Drop table TypeOfAttraction
Drop table PetsOnTrail
Drop table PetType
Drop table CommentsForTrail
Drop table RatingsForTrail
Drop table Trail
Drop table TrailDifficulty
```

```
Drop table Users
Drop table Park

--Create Tables in the following order
CREATE TABLE Park(
ParkID INT Identity(1,1) PRIMARY KEY NOT NULL,
ParkName VARCHAR(100) NOT NULL,
ParkAddress VARCHAR(300) NOT NULL,
ParkCity VARCHAR(100) NOT NULL,
ParkState VARCHAR(100) NOT NULL,
ParkZipCode INT NOT NULL,

CONSTRAINT UniquePark UNIQUE
NONCLUSTERED(ParkName)
)

CREATE TABLE TrailDifficulty(
diffID INT Identity(1,1) PRIMARY KEY,
difficultyWord VARCHAR(100) NOT NULL,
difficultyDescription VARCHAR(100) NOT NULL
)

CREATE TABLE Trail(
TrailID INT Identity(1,1) PRIMARY KEY,
TrailName VARCHAR(200),
TrailInParkID INT REFERENCES Park(ParkID) ON DELETE
CASCADE,
TrailLength INT NOT NULL,
Difficulty INT REFERENCES TrailDifficulty(diffID) ON DELETE
SET NULL,
GPScoords VARCHAR(100),
Constraint UniqueTrail UNIQUE NONCLUSTERED (TrailID,
TrailName, TrailInParkID)
)
```

```
CREATE TABLE Users(
userID int identity(1,1) PRIMARY KEY,
username varchar(50) NOT NULL,
FirstName varchar(20) NOT NULL,
LastName varchar(20) NOT NULL,
Pword varchar(75) NOT NULL,
CONSTRAINT UniqueUser UNIQUE NONCLUSTERED(username)
)

CREATE TABLE CommentsForPark(
ParkCommentID INT Identity(1,1) PRIMARY KEY,
CommentForParkID INT REFERENCES PARK(ParkID) ON
DELETE CASCADE,
CommenterID INT REFERENCES USERS(UserID) ON DELETE
CASCADE,
CommentText VARCHAR(255) NOT NULL
)

CREATE TABLE ParkRatingsTracker(
ParkTrackerID INT Identity(1,1) PRIMARY KEY,
RatedParkID INT REFERENCES Park(ParkID) ON DELETE
CASCADE,
RatingUserID INT REFERENCES Users(userID) ON DELETE
CASCADE,
Rating INT NOT NULL
)

CREATE TABLE CommentsForTrail(
TrailCommentID INT Identity(1,1) PRIMARY KEY,
CommentForTrailID INT REFERENCES Trail(TrailID) ON DELETE
CASCADE,
CommenterID INT REFERENCES Users(UserID) ON DELETE
CASCADE,
CommentText VARCHAR(255) NOT NULL
)
```

```
CREATE TABLE TrailRatingsTracker(
TrailTrackerID INT Identity(1,1) PRIMARY KEY,
RatedTrailID INT REFERENCES Trail(TrailID) ON DELETE
CASCADE,
RatingUserID INT REFERENCES Users(userID) ON DELETE
CASCADE,
Rating INT NOT NULL
)

CREATE TABLE TrailNotices(
    NoticeID INT Identity(1,1),
    TrailID INT REFERENCES Trail(TrailID) ON DELETE
CASCADE,
    noticeText VARCHAR(50) NOT NULL)

CREATE TABLE TypeOfAttraction(
TypeID INT Identity(1,1) PRIMARY KEY,
typeof VARCHAR(255) NOT NULL
)

CREATE TABLE Attractions(
AttractionID INT Identity(1,1) PRIMARY KEY,
AttractionName varchar(100) NOT NULL,
AttractionAddress VARCHAR(100) NOT NULL,
AttractionCity VARCHAR(100) NOT NULL,
AttractionState VARCHAR(2) NOT NULL,
AttractionZipCode INT NOT NULL,
AttractionDescription VARCHAR(100),
AttractionWebsite VARCHAR(100),
AttractionPhone VARCHAR(15),
AttractionType INT REFERENCES TypeOfAttraction(TypeID)
)

CREATE TABLE AttractionsNearPark(
```

```
NearParkID INT Identity(1,1) PRIMARY KEY,
NearAttractionID INT REFERENCES Attractions(AttractionID),
InParkID INT REFERENCES Park(ParkID) ON DELETE
CASCADE
)

CREATE TABLE PetType(
PetTypeID INT Identity(1,1) PRIMARY KEY,
typeText varchar(25) NOT NULL
)

CREATE TABLE PetsOnTrail(
    PetID INT Identity(1,1) PRIMARY KEY,
    PetOnTrailID INT REFERENCES Trail(TrailID),
    PetTypeID INT REFERENCES PetType(PetTypeID)
)

CREATE TABLE FundingForPark(
    FundID INT Identity(1,1),
    FundedParkID INT REFERENCES Park(ParkID) ON DELETE
CASCADE,
    FundName VARCHAR (75) NOT NULL,
    FundAddress VARCHAR(50) NOT NULL,
    FundCity VARCHAR(25) NOT NULL,
    FundState VARCHAR(2) NOT NULL,
    FundZipCode INT NOT NULL,
    FundDescription VARCHAR(255) NOT NULL,
    FundWebsite VARCHAR(75),
    FundPhone VARCHAR(15) NOT NULL
)

--Insert in the following order
INSERT INTO Park VALUES ('Hocking Hills State Park', '19852
State Route 664 S','Logan','Ohio',43138);
```

```
INSERT INTO Park VALUES ('Jackson Lake State Park', '35 Tommy Been Road','Oak Hill','Ohio',45656);
INSERT INTO Park VALUES ('Cleveland Lakefront State Park', '8701 Lakeshore Blvd, NE','Cleveland','Ohio',44108);
INSERT INTO Park VALUES ('Portage Lakes State Park', '5031 Manchester Road','Akron','Ohio',44319);
INSERT INTO Park VALUES ('Tinkers Creek State Park', '10303 Aurora Hudson Rd.','Streetsboro','Ohio',44241);
INSERT INTO Park VALUES ('Lake Milton State Park', '16801 Mahoning Avenue','Lake Milton','Ohio',44429);
INSERT INTO Park VALUES ('Buckeye Lake State Park', '2905 Liebs Island Road','Millersport','Ohio',43046);
INSERT INTO Park VALUES ('Mosquito Lake State Park', '1439 State Route 305','Cortland','Ohio',44410);
INSERT INTO Park VALUES ('Beaver Creek State Park', '12021 Echo Dell Road','East Liverpool','Ohio',43920);
INSERT INTO Park VALUES ('Deer Creek State Park', '20635 State Park Road 20','Mt. Sterling','Ohio',43143);
INSERT INTO Park VALUES ('Alum Creek State Park', '3615 S. Old State Road','Delaware','Ohio',43015);
INSERT INTO Park VALUES ('Blue Rock State Park', '7924 Cutler Lake Road','Blue Rock','Ohio',43720);
INSERT INTO Park VALUES ('Buck Creek State Park', '1901 Buck Creek Lane','Springfield','Ohio',45502);
INSERT INTO Park VALUES ('Caesar Creek State Park', '8570 E State Route 73','Waynesville','Ohio',45068);
INSERT INTO Park VALUES ('Hueston Woods State Park', '6301 Park Office Road','College Corner','Ohio',45003);
INSERT INTO Park VALUES ('Lake Loramie State Park', '4401 Ft. Loramie Swanders Road','Minster','Ohio',45865);
INSERT INTO Park VALUES ('Mohican State Park', '3116 State Route 3','Loudonville','Ohio',44842);
INSERT INTO Park VALUES ('Mt Gilead State Park', '4119 State Route 95','Mt. Gilead','Ohio',43338);
```

```
INSERT INTO Park VALUES ('Sycamore State Park', '4675 N.
Diamond Mill Road','Trotwood','Ohio',45426);
INSERT INTO Park VALUES ('Nelson-Kennedy Ledges State
Park', 'State Route 282','Nelson Township','Ohio',44065);
INSERT INTO Park VALUES ('Wingfoot Lake State Park', '993
Goodyear Park Blvd','Mogadore','Ohio',44260);

INSERT INTO TrailDifficulty VALUES('White Circle','Easiest');
INSERT INTO TrailDifficulty VALUES('Green Circle','Easy');
INSERT INTO TrailDifficulty VALUES('Blue Square','More Difficult');
INSERT INTO TrailDifficulty VALUES('Black Diamond','Very
Difficult');
INSERT INTO TrailDifficulty VALUES('Dbl. Black
Diamond','Extremely Difficult');

INSERT INTO Trail Values('Mountain Bike',1,2,1,NULL)
INSERT INTO Trail Values('Mountain Bike',2,5,3,NULL)
INSERT INTO Trail Values('Mountain Bike',3,7,4,NULL)
INSERT INTO Trail Values('Park Office trail',4,1.5,1,NULL)
INSERT INTO Trail Values('Multipurpose',5,7,4,NULL)
INSERT INTO Trail Values('Bridle trails',6,38,2,NULL)

INSERT INTO TrailNotices VALUES(1, 'Closed')
INSERT INTO TrailNotices VALUES(2, 'CAUTION: Landslides')

INSERT INTO TypeOfAttraction VALUES('Lodging')
INSERT INTO TypeOfAttraction VALUES('Camping')
INSERT INTO TypeOfAttraction VALUES('Recreation')
INSERT INTO TypeOfAttraction VALUES('Food')

INSERT INTO Attractions VALUES('Longhorn Steakhouse', '111
Washington Blvd.','Wooster','OH', 44691,NULL,'http://
www.longhorn.com', '(330) 555-1234', 4)
```

```
INSERT INTO Attractions VALUES('Swensons', '24 Howe Ave','Akron','OH', 44245,NULL,'http://www.swensons.com','(330) 555-2345', 4)
INSERT INTO Attractions VALUES('Putt n Stuff', '53 Acre Drive','Wooster','OH', 44691, NULL, NULL,'(330) 555-3456', 3)
INSERT INTO Attractions VALUES('Mrs Millers Cabin', '455 State Route 557','Charm','OH', 44623, NULL, NULL,'(330) 555-3456', 1)
INSERT INTO Attractions VALUES('Camp Toodik', '5601 State Route 39','Loudonville','OH', 44605, NULL, NULL,'(330) 555-3456', 2)

INSERT INTO AttractionsNearPark VALUES(1, 2)
INSERT INTO AttractionsNearPark VALUES(4, 5)
INSERT INTO AttractionsNearPark VALUES(1, 6)
INSERT INTO AttractionsNearPark VALUES(1, 1)
INSERT INTO AttractionsNearPark VALUES(2, 1)
INSERT INTO AttractionsNearPark VALUES(3, 1)
INSERT INTO AttractionsNearPark VALUES(4, 1)
INSERT INTO AttractionsNearPark VALUES(5, 1)

INSERT INTO PetType VALUES('Only Dogs')
INSERT INTO PetType VALUES('Only Horses')
INSERT INTO PetType VALUES('No Pets')
INSERT INTO PetType VALUES('Dogs and Horses')

INSERT INTO PetsOnTrail VALUES(1,1);

INSERT INTO FundingForPark VALUES (1, 'Old Foagies Cave Ltd.', '123 Foagie Drive','Logan','OH','44444', 'This is a test fund.','http://www.oldfoagiesofhockinghills.org','(740) 555-1234')

--Select statements
select * from Park
select * from TrailDifficulty
select * from Trail
```

select * from Users
select * from CommentsForPark
select * from ParkRatingsTracker
select * from CommentsForTrail
select * from TrailRatingsTracker
select * from TrailNotices
select * from TypeOfAttraction
select * from Attractions
select * from AttractionsNearPark
select * from PetType
select * from FundingForPark

3.2. Creation Requirements

3.2.1. Hardware

Memory: 1.5 GB
Processor Type: 64 Bit
Processor Speed: 1.83 GHz
Hard Disk Space: 5 GB

3.2.2. Software

Operating System: Windows Vista 64 bit or later
Mac OS X or later

Web Browser: Internet Explorer 7 or later
Mozilla Firefox 3.5 or later
Apple Safari 5.0 or later
Google Chrome 1.0 or later

3.3. Failover

Using Microsoft SQL Server 2005, the option used for failover is database mirroring. This technology transfers transaction log record directly from one server to another and can quickly fail over to the standby server. Code is inserted into the client application to automatically redirect their connection information, and in the event of a failover, automatically connect to the standby server and database with the permission of a third server or "witness server" which works as a heartbeat between the primary and mirrored server. Database mirroring allows for a quick failover with no loss of committed data and does not require proprietary hardware and allows easy set up and management.

3.4. Backup and Recovery

Along with using database mirroring as an option for backup, a full backup will be implemented on Microsoft SQL server 2005 every Sunday. The primary database will be backed up onto at least 500 gigabyte external hard drive. Using both strategies, this will ensure no loss of information, at least on a weekly basis. Recovery will be done by switching to the mirrored database or by extracting the database files from the back up external hard drive.

3.5. Data archival

There is no plan for data archival because there is currently no use for archiving data. There are no instances of old or not current data that would need to be stored on tape or disk. In future plans, a strategy may be developed in order to archive the user's comments or ratings, but at this point there is no current use to archive any of this data.

3.6. Security requirements

The front end application is able to log in a user with and encrypted password linked with a user name. This session is unique for any user who has registered on the database through the website previously. This user name and password is unique and cannot be accessed by any other user. This process preserves the integrity of the database and does not allow any user to hack into another user account.

3.7. Beta Test Plans

1. **Load Testing**-These are the limits of the web server. There are requests to the HTML page every 2 seconds in a ramp test from 0 to 600 users. From the chart, the test went well until about 350 simultaneous users. Hits per second went up to 550 hits per second, but could not get any more clicks from this webserver while there were increased the number of users further to 600users.

2. **Stress Test**- In this stress test up to 500 users perpetually access the parks.php script from the web server without a click delay/think time (the users reload the page immediately after the last request is finished).
This chart shows the "Spectrum of Click Times" which shows the results of this Ramp Test. The three axis are:

- Vertical: percentage of users
- Horizontal: user wait time
- Depth: Number of users

With more and more users accessing the server the request times deteriorate, the bar's maximum is moving from left to right with increasing depth.
With more than 150 users, the wait time to load the page per user is between 2 and 5 seconds (see the red line in the chart).

3. **Performance Testing** – Overall performance of system is average. There are no complicated SQL queries to slow down the system, but the amount of traffic on the web application may slow down in high volumes. The performance of this system is average given the server type and speed of the average user computer. As long as the flow of traffic is not above 500-600 users at one time, the system will be able to perform at an optimal level.

4. **User-Acceptance Testing**- Out of a selected 10 users the front end application was overall rated as “excellent” and “very easy to navigate.” Overall, end users had no problem being able to navigate the website and were able to filter out needs according to trails, attractions, parks and city. This outcome is a result of the trails and attractions filter found via top-left of the screen and also hyper-links found at the top of the web page. These results may change over time with the addition of registered users.

3.8. Issues and Limitations

One issue is maintaining the front end application. There would either have to be a person or a team manually updating the web site if new comments, ratings, or attractions would be added to the database. There would also

have to be an administrator position in order to monitor the comment section of the website, so nothing vulgar is posted. Additionally, there is the constant need to keep all information up to date and accurate for users (i.e. Notices, Park Contact Information, etc). There are no current limitations to the database or the web application, but the issue of maintenance is an ongoing dilemma.