

Software Design Document Final

Challenge Expeditions

May15-15

Client: Michael Dalhstrom

Advisor: George Amariuca

Andrew Wallace - Project Leader

Anthony Wilson - Webmaster

Ian Monroe - Communications

Matthew Burzinski - Key Concept Holder

Table of Contents

Problem statement.....	3
Design Overview.....	3
Goals.....	3
Design Rationale.....	4
Use Cases.....	4
Starting the App.....	4
Finding a Nearby Park.....	6
Searching For a Specific Park.....	7
Starting a Challenge.....	8
Completing a Challenge.....	9
Building a Challenge.....	10
Creating an Account.....	11
Use Case Diagram.....	12
System Architecture.....	12
Mobile Client.....	12
Server.....	13
Database.....	13
Architecture Diagrams.....	14
Data Design.....	14
User.....	14
Challenge.....	15
Database Schema.....	17
Technology Rationale.....	17
Android Framework.....	18
PHP Web Server.....	18
MySQL Database.....	19
Interface Design.....	20
User Interface Description.....	22

Test Plan.....	22
Conclusion.....	23

Problem Statement

We as a society aren't as active as we use to be. The amount of time we spend indoors and on our cell phones is alarming. We want to encourage exploration and activity with a new application. We'd like to combine the technology we use everyday to inspire activity, challenges and natural exploration. We also want to create a way to share these experiences with others to promote friendly competition and others to participate.

Design Overview

The biggest challenge we face designing our project is building a system that allows us to reduce redundancy and provide an adaptable interface for expansion and maintainability. With a limited amount of time, need to limit the amount of code we have to write. We also want the system to provide a way to add on new features and changes. If we can provide a way to abstract sensor data we'll able to add new sensors and create challenges that use those sensors very easily.

Goals

- Reduce redundancy
- Highly scalable
- Easy maintainability

Design Rationale

The system works by separating implementation into mobile client, server (web application and API), and database modules. We decided to go with this model because they align with our goals. In terms of reducing redundancy we focus on building a mobile application that uses an API for everything it needs. This allows us to only have to write specific calls once on the API and let the web application and mobile application use those calls. It also means when something needs to change, we just change it in the API and everything just works.

We also separated the modules in such a way, that they can easily be swapped out and replaced. We want them to be highly decoupled and very much replaceable. If this app is to be scalable and be capable of advancing, there is a high chance that performance would need to be increased and having more replaceable modules will expedite this process. This will also allow easier updates and overall code maintenance better.

Use Cases

1. Starting the App

- a. Actors
 - i. User of the App
 - ii. Online Server
- b. Preconditions
 - i. App has been successfully installed
 - ii. Phone has internet access
 - iii. User has a profile created
- c. Basic flow of Events
 - i. User starts the app by clicking the icon on her phone
 - ii. App goes full screen and begins loading

- iii. Login Screen is open
- iv. User clicks on Input box for username or password
- v. App brings up the user's preferred keyboard
- vi. User inputs her username or email and password
- vii. User clicks login or hits the enter key on the phone
- viii. USE CASE: Validate User is performed
- ix. User logs in successfully
- x. App displays main menu, ending the use case

d. Alternative Flows

i. Invalid User

1. On step viii, if the username is not recognized, use case fails and "Username not found" displays, goes to step iii.

ii. Invalid Password

1. On step viii, if the password does not match the password for the given username, use case fails and "Incorrect Password" is displayed, goes to step iii.

iii. Create a new account

1. On step iv, users may instead opt to create a new account by pressing a "sign up" button
2. Perform Use Case: Creating an Account
3. Use case ends successfully

iv. No internet Access

1. On step iii, if no internet access can be found, app prompts user "No internet found, start in offline mode? Yes | Cancel | Try Again"
 - a. Hitting Cancel closes the app
 - b. Hitting Try Again attempts to connect to the internet and goes to step iii.

- c. Going in offline mode starts Use Case: Offline Mode start

2. Finding a Nearby Park

- a. Actors
 - i. User of App
 - ii. Online Server
 - iii. Database of Parks
 - iv. Google Maps or similar system
- b. Preconditions
 - i. User has successfully logged into the app
 - ii. Phone has internet access
 - iii. Phone has access to a maps system
- c. Basic Flow of Events
 - i. From the home page (landing page from login), user clicks on the "Find Nearby Parks" Option
 - ii. App accesses phone's gps coordinates
 - iii. Sends the coordinates to the online Server
 - iv. Online Server searches through the database and finds the (5) closest parks
 - v. Online Server sends simple data of the parks to the app
 - vi. App displays the parks sorted by nearest-park-first
 - vii. User can click on one of the parks and another request is sent to the server
 - viii. This time the server will send back the full park data
 - ix. Use case ends, sending user to the park details page
- d. Alternative Flows

- i. No Internet Access
 - 1. Page simply displays “No internet connection can be found”
 - 2. button “try to establish connection?”
 - 3. pressing the button attempts to find a connection, goes to (ii)
- ii. No GPS
 - 1. Page displays “No GPS Signal can be found”
 - 2. button “try to establish a signal?”
 - 3. pressing button attempts to find a signal, goes to (ii)

3. Searching for a specific Park

- a. Actors
 - i. User of App
 - ii. Online Server
 - iii. Online Database of Parks
 - iv. On Phone Database of Parks
- b. Preconditions
 - i. User has successfully logged in
 - ii. Phone has internet or downloaded database of parks
- c. Basic Flow of Events
 - i. From the landing page, User clicks “Search Parks”
 - ii. Search parks page comes up, with a list of the park data downloaded.
 - iii. User can type in a park name or city and hit “search”
 - iv. Phone first checks through any downloaded parks data and displays that
 - v. Phone sends request to online server
 - vi. Online Server performs a search on the database
 - vii. Sends back simple data for listing purposes

- viii. Phone displays search results with limit (5)
- ix. User can click on a result and another request is sent to the server
- x. This time the server will send back the full park data
- xi. Use case ends, sending user to the park details page
- d. Alternative Flow of Events
 - i. No Internet Access
 - 1. Behaves as normal, but no online server requests are sent
 - 2. searches only go through downloaded database
 - 3. Details of parks only come from the downloaded database
 - ii. No Park Found
 - 1. Page displays “ no results found, try broadening your search”

4. Starting a Challenge

- a. Actors
 - i. User of App
 - ii. Park Details Data
 - iii. Online Server
- b. Preconditions
 - i. User has successfully logged in
 - ii. Internet Connection available
 - iii. Any necessary features specific to the challenge are on the phone
- c. Basic Flow of Events
 - i. User clicks a challenge from the park details page
 - ii. Challenge details page comes up

- iii. App determines whether phone has necessary features for the challenge
 - iv. User starts the challenge
 - v. App records starting conditions of the challenge
 - vi. Challenge is added to the Active Challenges list
 - vii. Specific Challenge Protocol is followed
- d. Alternative Flow of Events
- i. Required Features Not Found
 - 1. Start button is greyed out and inaccessible
 - 2. Message "Features necessary for this challenge cannot be found"
 - 3. "Features needed: " and then a list of what is missing

5. Completing a Challenge

- a. Actors
 - i. User of App
 - ii. Park Details Data
 - iii. Online Server
- b. Preconditions
 - i. User has successfully logged in
 - ii. Internet Connection available
 - iii. Any necessary features specific to the challenge are on the phone
 - iv. User is in a park
 - v. Challenge exists
- c. Basic Flow of Events
 - i. User has started a challenge
 - ii. User follows instructions given on the phone
 - iii. Phone tracks when parts of challenge are completed
 - iv. User inputs when challenge is complete

- v. Phone confirms based on gps tracking that challenge is complete
- vi. System updates user statistics and leaderboards =
- d. Alternative Flow of Events
 - i. Required Features Not Found
 - 1. Start button is greyed out and inaccessible
 - 2. Message "Features necessary for this challenge cannot be found"
 - 3. "Features needed: " and then a list of what is missing
 - ii. GPS data says challenge is not complete

6. Building a Challenge

- a. Actors
 - i. User of App
 - ii. Park Details Data
 - iii. Online Server
- b. Preconditions
 - i. User has successfully logged in to mobile or web application
 - ii. Internet Connection available
- c. Basic Flow of Events
 - i. User selects location for the challenge
 - ii. User names challenge
 - iii. User selects the type of challenge
 - iv. User inputs challenge parameters
- d. Alternative Flow of Events
 - i. Required Features Not Found
 - 1. Start button is greyed out and inaccessible
 - 2. Message "Features necessary for this challenge cannot be found"
 - 3. "Features needed: " and then a list of what is missing

7. Creating an Account

- a. Actors
 - i. User of App
 - ii. Online Server
- b. Preconditions
 - i. Internet Connection available
 - ii. User Downloads application
- c. Basic Flow of Events
 - i. User inputs username and password
 - ii. User inputs information
 - iii. User uploads profile picture
- d. Alternative Flow of Events
 - i. Required Features Not Found
 - 1. Start button is greyed out and inaccessible
 - 2. Message "Features necessary for this challenge cannot be found"
 - 3. "Features needed: " and then a list of what is missing
 - ii. Username or password is invalid
 - 1. User inputs new username and password

Use Case Diagram

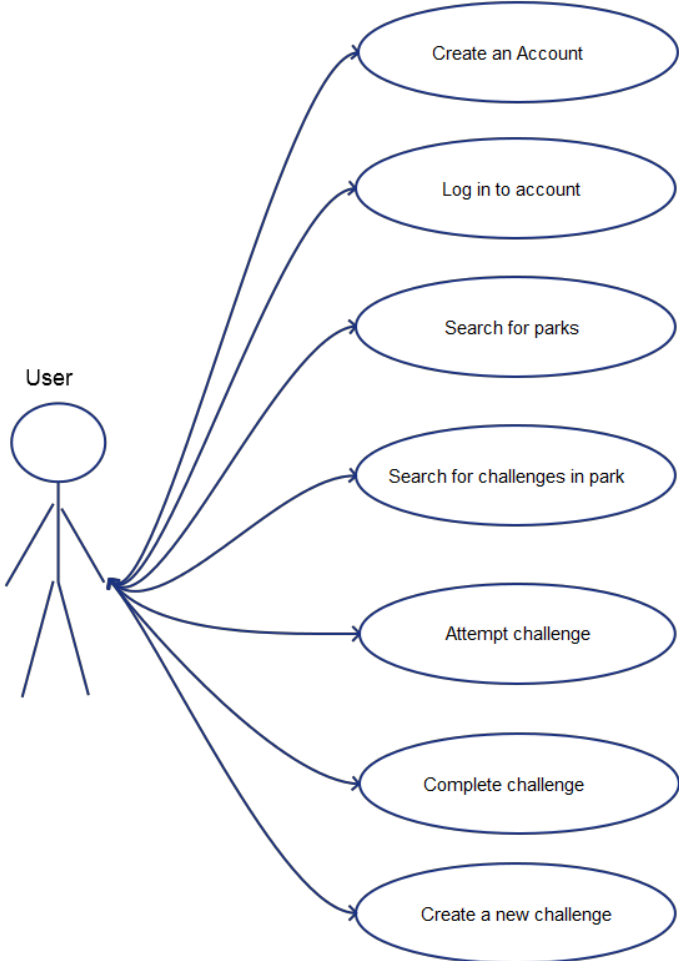


Figure 1: Options for the User

System Architecture

Mobile Client

The mobile client is responsible for providing the everyday interface and utilities for the user. It will be composed of breaking down the sensor data and registering it with the challenges and activities. The application must provide a usable and

understandable interface as this will be the primary device for the system. The mobile client is responsible for accessing local and remote sensor data. It must verify the completion of challenges and sync the data with the server. Along the same lines, it should be able to pull challenges and stats for challenges, friends, and parks. It should be able to find new challenges and parks as well.

Server

The server will be the interface for the web application and the API. The web application will provide a limited number of features that the mobile client has. These features are viewing stats on challenges, managing profiles, and finding parks. The user will not be able to perform challenges through the site or access sensor data in any way.

The server also holds the API for our system. The API provides all the calls for accessing any data about challenges or profiles. Any outside data that we need to access (Google Maps, park and recreation data, etc.) will also be accessed through our API and will fetch the data. It also allows storing of data that needs to persist in a central way.

Database

The database will simply store application data. It will be built on the fundamental models of our system (challenges, users, and badges).

Architecture diagrams

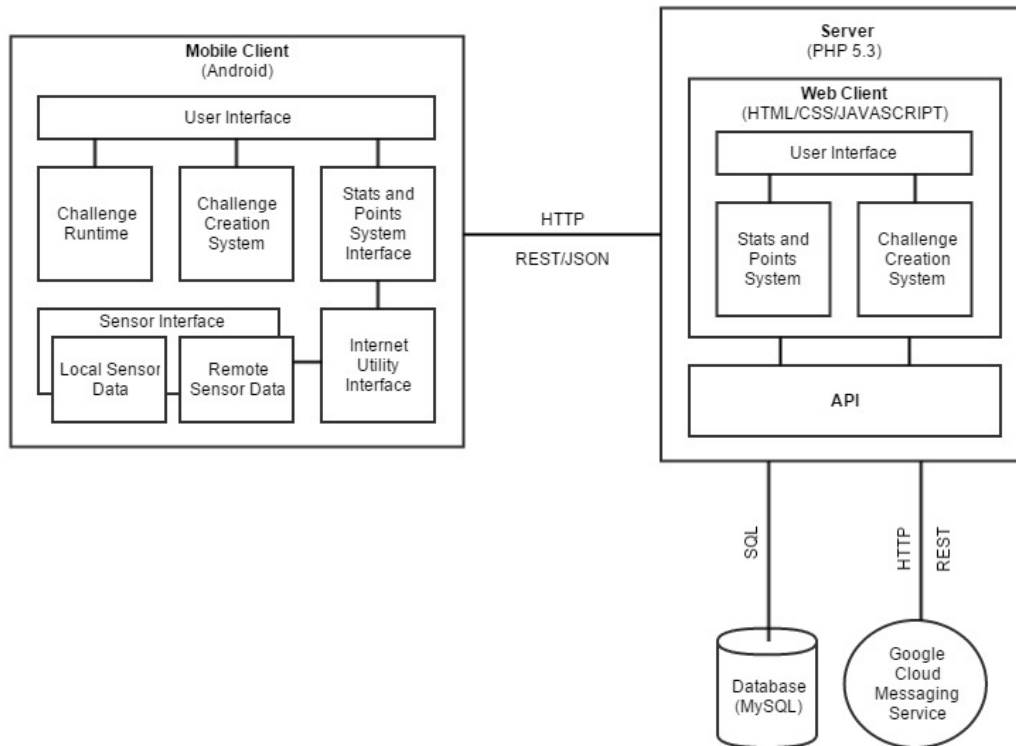


Figure 2: Basic System Architecture

Data Design

User

User will contain information involve the following:

1. Personal Information
 - a. Name
 - b. Age
 - c. Gender
 - d. Username
 - e. Password
 - f. Friends
 - g. Photos
2. User Application Information
 - a. Parks visited

- b. Challenges completed
- c. Scores

Challenge

A challenge is the most important data model for our application. Below we have some of the types of challenge we would like to implement.

1. Running/Biking- Race type (get to finish as quick as possible, any route)

- 1.1. *Associated Park*
- 1.2. *Start Location*
- 1.3. *End Location*
- 1.4. *Maximum Avg. Speed (Prevent cheating)*
- 1.5. Player's Avg. Speed
- 1.6. Players Current Location
- 1.7. Players Start Time
- 1.8. Players End Time
- 1.9. Weather at time of race
- 1.10. Points are given for avg. speed times distance of race.
- 1.11. Bonus points given for improving one's score
- 1.12. Bonus points given for top 10 in leaderboard
- 1.13. Bonus points given for Extreme Weather

2. Running/Biking - Free Ride

- 2.1. *Associated Park*
- 2.2. Players Speed (only to determine distance)
- 2.3. Weather
- 2.4. Points given for Distance Only
- 2.5. Bonus points for improving one's own score
- 2.6. Bonus points for Extreme Weather
- 2.7. No Leaderboard

3. Running/Biking - Checkpoint (Players must go through specific checkpoints before finish)

- 3.1. *Associated Park*
- 3.2. *Start Location*
- 3.3. *Ordered list of next locations*
- 3.4. *max avg. speed(prevent cheating)*
- 3.5. Players Current Location
- 3.6. Players Start time
- 3.7. Players End Time
4. **Running/Biking - Time Attack (Need to reach a gate in limited time, time gets added on for each gate. Go through as many gates as possible)**
5. **Running/Biking - Endurance (Go as long as possible without ever going below a certain speed)**
6. **Biking/Running - Collection (Get to checkpoints in any order)**
7. **Biking - Speed Trap (Similar to Checkpoint, but speed is recorded at each point and added)**
8. **Biking - Zone (after set time, need to maintain a higher and higher min. speed)**
9. **Orienteering (Map checkpoints, uses clues)**

Database Schema

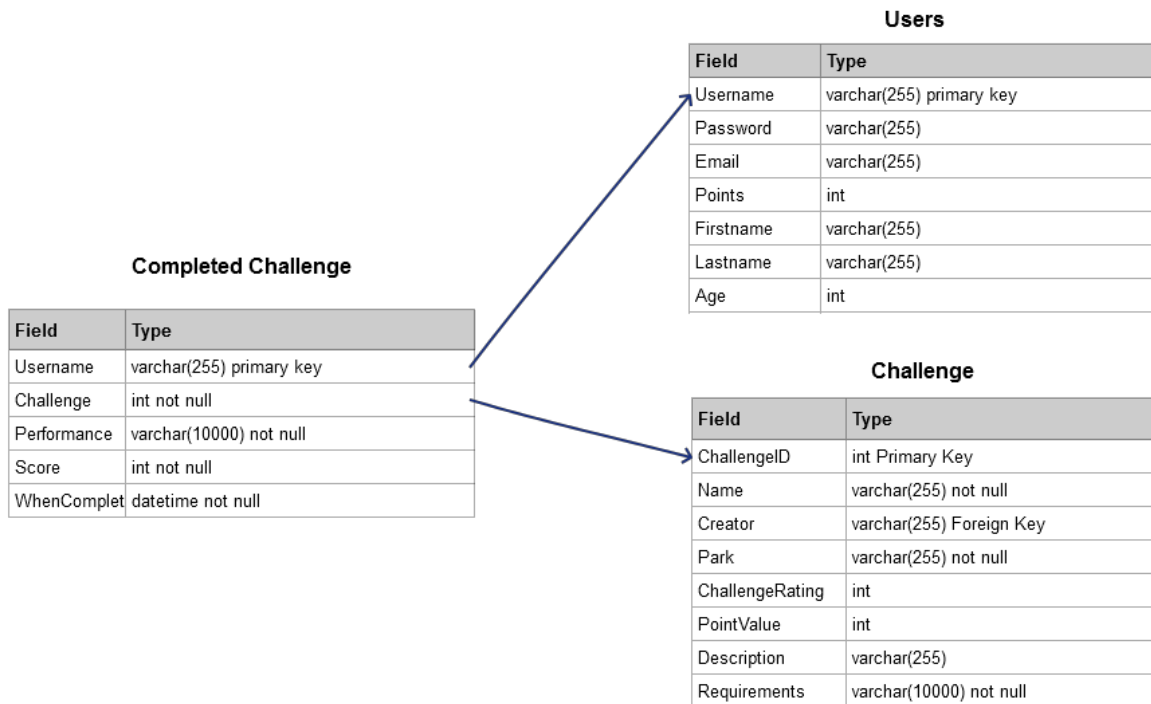


Figure 3: Initial Database Design

Technology Rational

The project we are trying to accomplish requires multiple tools to be successful. We spent lots of time doing research on mobile application frameworks, web languages and tools, and databases. When doing this research, we really had to consider what were the most important factors in getting this project accomplished well.

When we thought about it, and time was the biggest factor. We only have two semesters with all of us having classes and work that take up our time. We need to be able to start quick and change things easily. Familiarity with tools helps speed up thing because we do not have to learn new tools. We also need to be able to

prototype quickly because not many of us are familiar with the domain for the project. Quick prototyping allows us to make mistakes and fix them rapidly.

The other important factor is scalability. We want this app to be expandable in the future. We need to ensure the technologies we use can accommodate that with future support and decent user base stability. Choosing to work with frameworks and languages that cater to future growth is something we value.

Android Framework

We originally decided to go with Mono for Android. We thought it would save more time by having the same code base in C#. This would have allowed us to build the iOS application fast when we are done with the prototype. We were a bit hesitant because it's in a different language than most of us know but since it's similar to Java we think it won't be too difficult to transition.

When using Mono it had a standard IDE called Xamarin that went hand in hand with it. As we got further into our project we realized that Xamarin only allowed us to create projects of a certain size while using the free version. Instead of paying the fee we decided to switch to using the standard Android Application Framework and Android Studio as the IDE. Although it was a bit of a setback because we had to scrap most of the work we had done using mono, the transition was pretty easy because many of us have used Android studio and the Android Framework before.

PHP Web Server

We decided to make the web application and API using PHP. We believe it was a good choice due to its great developer community and easy learning curve. It also helps that at least one person on our team has already used PHP before. We'll be able to ask for help and easily find resources for what we want to do.

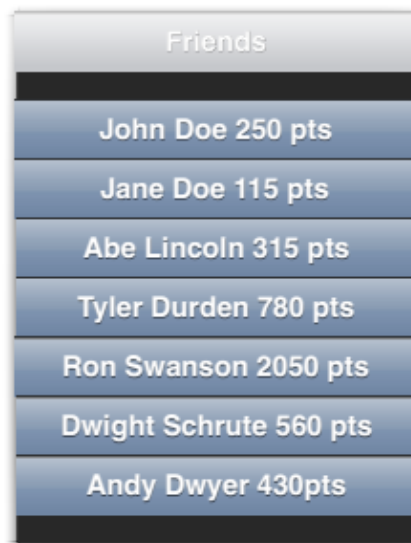
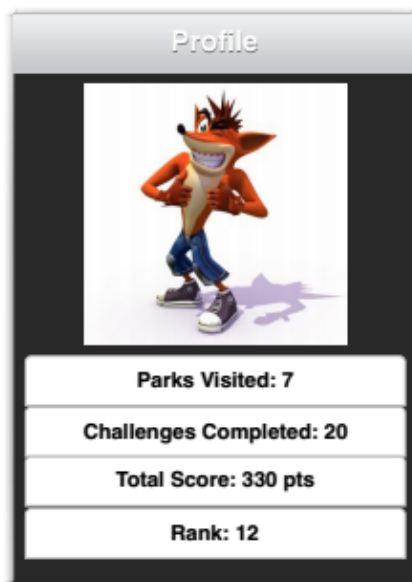
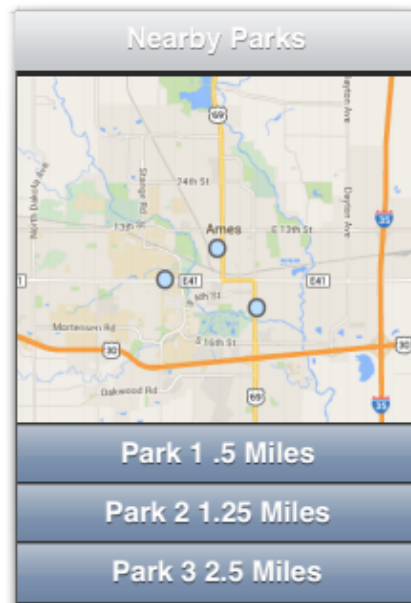
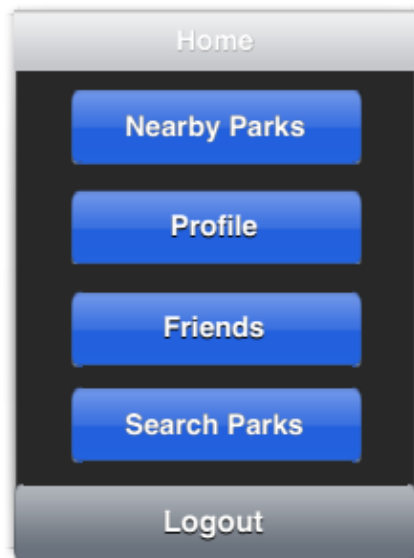
We considered Java and Spring MVC because we all know Java, but we felt that configuration would be a hassle and writing APIs in Java is clunkier and not as quick to prototype.

MySQL Database

We mostly chose MySQL because it was what a lot of our Iowa State projects use as a database. It's fast, simple, and efficient. A NoSQL database probably wouldn't have been a bad idea as our data models might change often during the prototyping phase. We decided not to go with a NoSQL database just because we're familiar with SQL mostly and PHP pairs nicely with SQL as it is.

Interface Design

Mobile application





Figures 4-10: Screen Concepts for the App, labeled at top of each screen

User Interface Description

Mobile application flow

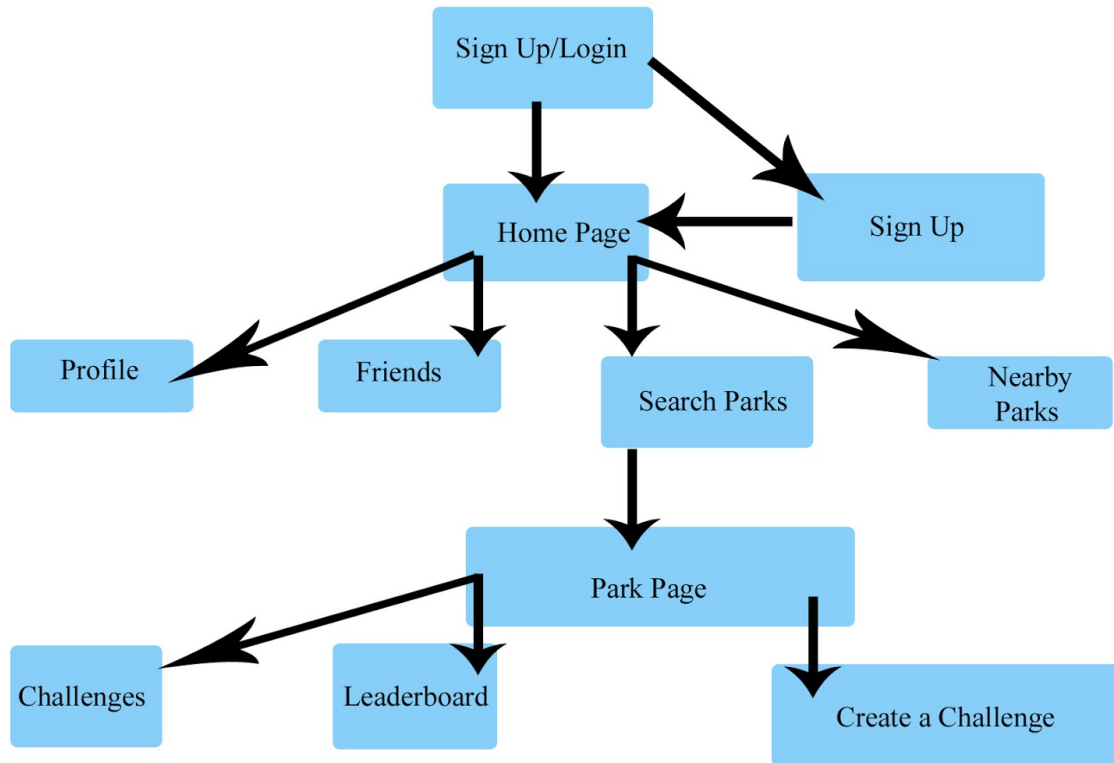


Figure 11: Page Flow Diagram

Test Plan

We decided to put a focus on regression testing and unit testing. We believe that our application provides many individual parts which will require us to ensure components behave as expected. Unit testing throughout the building process will help maintain workable, modular code. The work environments that we plan to use natively allow unit testing along with built-in tools and integration. This is a benefit that we believe will help us in the long run which also factored into our decision.

We believe thorough regression testing will allow us to build upon working code and ensuring that we don't break anything as we add features. We expect rapid changes in our application but would like to have the confidence that our code works even after those changes. Regression testing and tools will allow us reach that goal. In addition, setting up proper regression testing is important as this is an application that is expected to grow, even once it goes outside the scope of senior design.

Conclusion

We believe this application is an exciting way to get people outside and having fun. With the proper planning and preparation the project can be accomplished and fulfill all the goals stated above. It's very much feasible and we are excited to create a prototype to further explore the possibilities of this application and show the full potential of the idea. The application has a huge potential growth factor, and it will be interesting to see how popular this app can become.