

Final Report

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
Deliverables.....	3
First semester.....	3
Second semester.....	3
Functional requirements.....	4
Non-functional requirements.....	4
Specifications.....	4
Concept sketch/mockup.....	4
User interface description.....	6
Work breakdown structure.....	7
Project roles.....	7
Individual roles.....	7
Test Plan.....	8
Resource requirements.....	8
Project schedule.....	9
Risks.....	9
Market literature/survey.....	10
Conclusion.....	10

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.

Deliverables

First Semester

The goal of the first semester was to create a prototype of the application. To get there we need to create a solid plan and a design that will allow us to build a solid and scalable prototype. We want to get basic functionality done so we can build upon that in the second semester.

- Strong project plan
- Feasible design for the application and server
- Android application prototype
 - User profiles creatable on app and web
 - 2 sensors functional
 - 2 challenges can be ran and verified
 - Can view completed challenges on app and web
 - Web API support all conditions above

Second Semester

The goal for the second is to polish and build upon the prototype. We will do that while concurrently building an equivalent iOS app. Much of the extras features and use cases will be accomplished in this semester.

- Challenge creation system implemented
- Equivalent iOS, Android and Windows Phone applications

- Push notification system
- Completed and extendable web API

Functional Requirements

- Create/login/logout of profile
- View completed challenges/points/badges (etc)
- View new challenges to complete
- Share challenge with another user
- Run and verify challenge to completion
- Use all available sensors on device
- Upload photo/videos to challenges
- Offline mode for when service can't be found

Non-Functional Requirements

- 95% server uptime
- Uploaded challenges viewable in less than 3 mins
- Always responsive application
- API response in less than 30 secs
- Scalable

Specifications

- Cross-Platform: should be available and fully functional across all web browsers and current Android/iPhone devices
- Database: Store data related to all state parks, all users, and all challenges
- Challenge Framework: Easy to understand, create, and share challenges.
- Server: Accurate and fast completed challenge verification. Handle all database operations and push-notifications in a timely manner.

Work Breakdown Structure

Project Roles

- Andrew Wallace - Project Leader
- Anthony Wilson - Webmaster
- Ian Monroe - Communications
- Matthew Burzinski - Key Concept Holder

Individual Work Roles

- Andrew Wallace - Mobile application screens and framework, device testing, sensor data
- Anthony Wilson - Database management, Server challenge verification/management
- Ian Monroe - User profile system, Challenge stats API, Push notification system
- Matthew Burzinski - Application Design, Front-end, Application-Side technology, dividing work into small stories/tasks

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 requires 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.

Resource Requirements

Resource	Will we get it?	Estimated cost
Server	We will ask the client for a database and a web application server to handle the web app and API.	\$0
Iowa Parks Information	We will ask client first. If unavailable by the client we will contact the Iowa Parks and Recreation Association.	\$0
Jira Task Management	If we have enough server space we will purchase Jira from Atlassian.	\$10
iOS Developer License	Depending on how much we would like to test on hardware vs emulator, we will need a license	\$99

Table 1. Table outlining expected cost for required tools.

Project Schedule

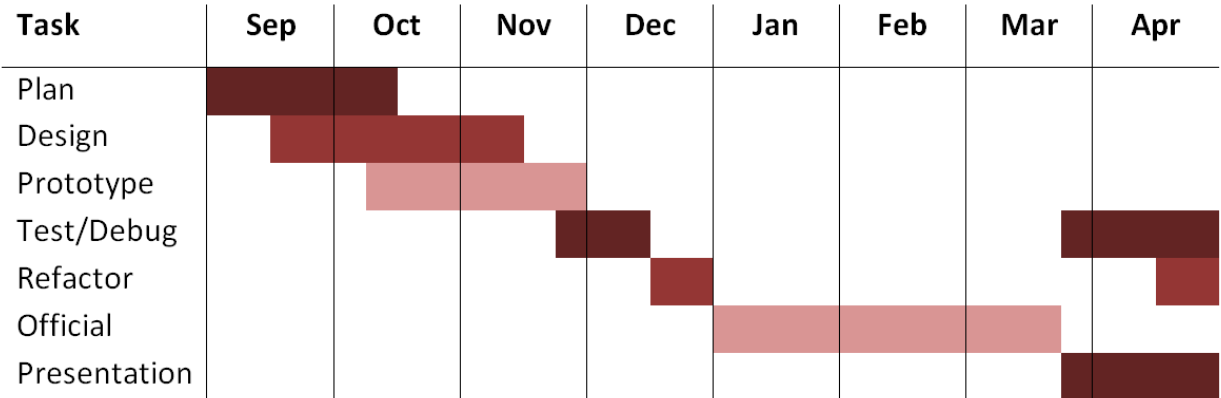


Table 2. Expected schedule for school year and project.

Risks

One of the risks we have with this application is cell phone reception and data connection. Our intended use is to be used in parks and places where data connection isn't as available. We will need to learn how to handle those particular situations and provide alternative methods of functionality.

Some challenges we plan on testing can become complicated in how the user may store their phone or use the application. For example, say we want a swimming challenge. How does the user swim with their phone? Is it our responsibility to recommend options or to warn of risks? These are questions we have to ask for a variety of different circumstances and challenges.

Along the same lines with complex situations, we also have many other weird problems when it comes to verifications of challenges. How can we ensure a user camped out all night at a local state park? How do we determine if a user really ran those 10 miles? Do we remove stats and competition if we can't ensure fairness? These questions arise when we talk about validating challenges and will be something we have to figure out.

Market/Literature Survey

Today it is hard to find someone without a smartphone or easy access to the internet. It is much harder to find people engaged in outdoor adventure. We believe it is human nature to be adventurous, and that deep down, people want adventure. With this application we believe we can make modern technology a tool for people to engage in outdoor adventure, and make it a social event.

Everyone is constantly uploading pictures of their "adventures" on social media platforms. The lunch they had today, the shoes they bought, the show they watched. We think that by merging this social aspect of sharing adventure into our application we can reach a broad range of people. We will enable people to engage in adventure socially, and further increase participation by means of our challenge system. By allowing users to challenge others to a new adventure, or even

adventures they've already completed, we should spark the natural adventurous fire that is lingering in a wide range of people.

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

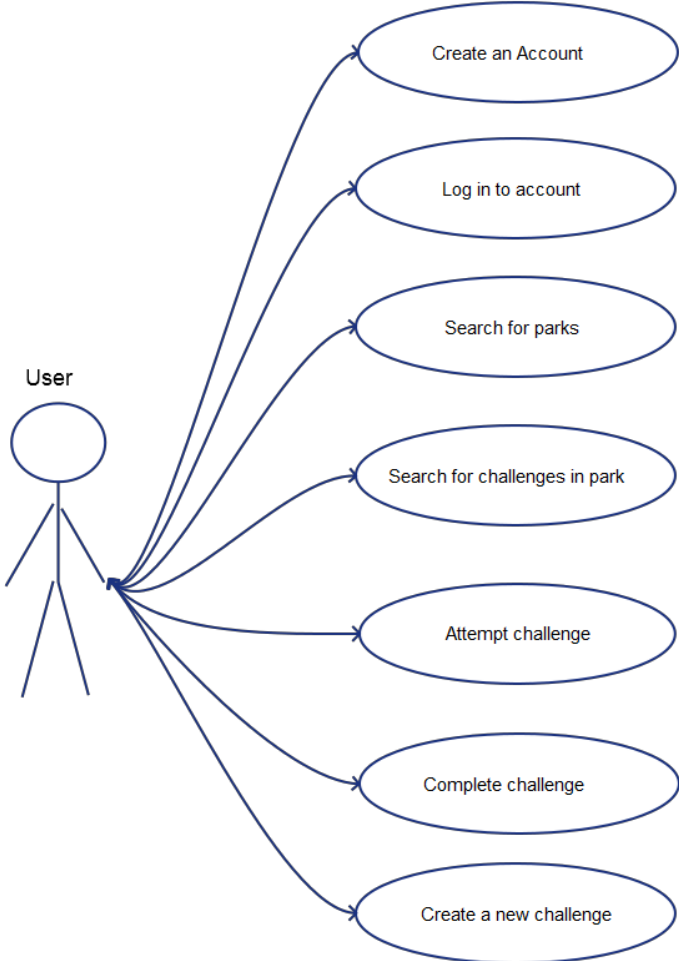


Figure 1: Options for the User

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

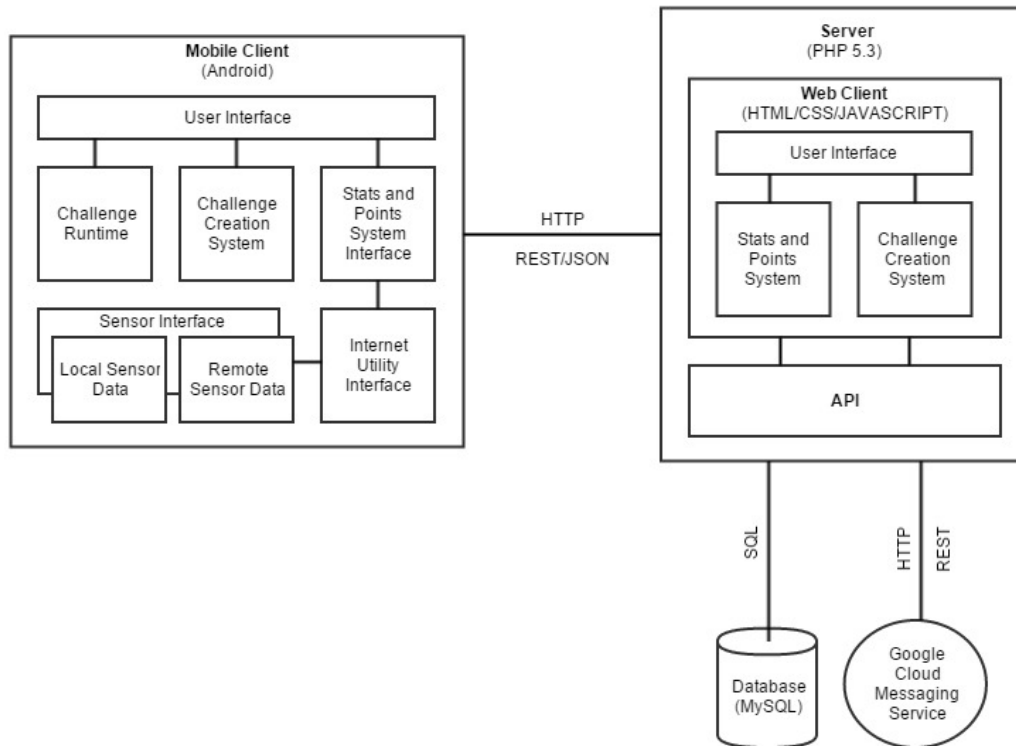


Figure 2: Basic System Architecture

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

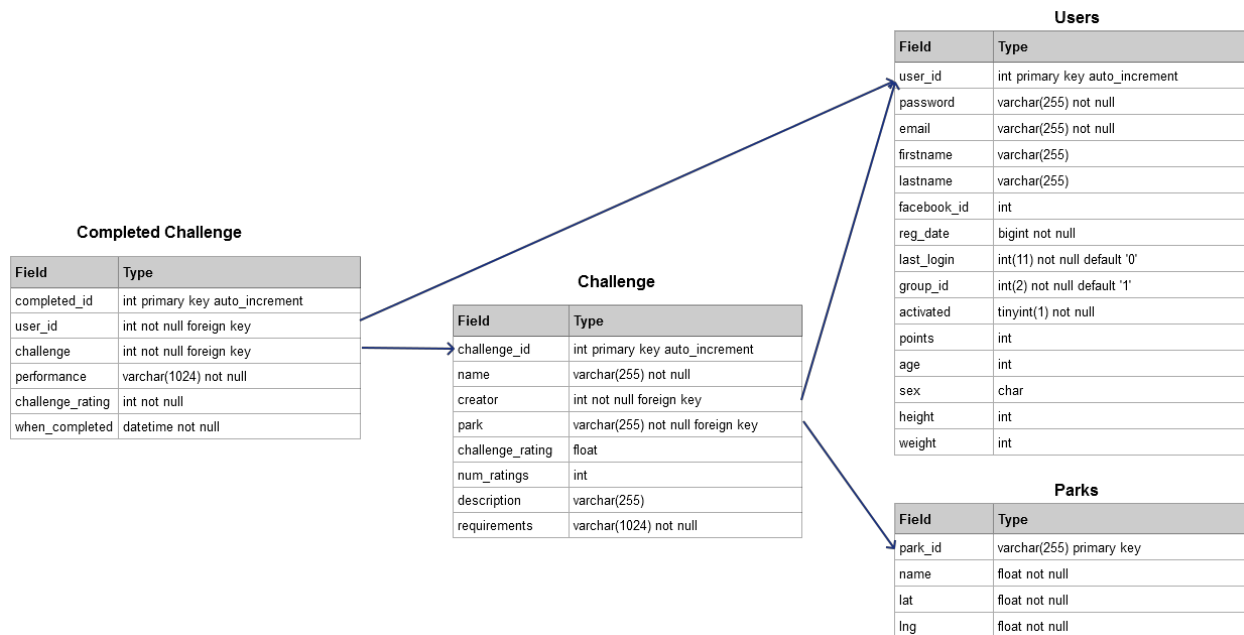


Figure 3: Initial Database Design

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.

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.

Appendix I

User Manual

Mobile Application

Downloading The App

Go to the Google Play Store and search for Master Explorer. This can be found here: _____ (App has not been submitted to the play store yet, currently need to contact us to get a download)

Signing On

When you open the app for the first time, you will be greeted with a page asking you to login. If you have already registered through the website, or this is not your first time using the app, then simply give your email and password and press the "Log In" button, otherwise press the "Register" button. Registering is easy, simply fill out the fields for your first and last name, email, and password, then press "register".

Navigating the App

Upon logging in, you will first be brought to your profile page. In the upper left corner there is a button that will open a slide menu. From here, you can get to the five main pages of the app. Other pages, such as the page for running a challenge, can be accessed from these pages when appropriate. The slide menu also has an option for logging out.

Profile

The profile page contains a picture if you've uploaded one on the website, your total points from completing challenges, the highest number of points you've received from a single challenge, and the point value of the last challenge

completed. Underneath these is a list of all the challenges you have completed. These challenges can be clicked on to retake the challenge, or just access the challenge screen.

Challenge

The challenge page shows the title of the Challenge, a picture associated with the challenge, and the tasks associated with the challenge. The tasks show the starting location, the distances, times, temperatures and the ending location. You can also click "start" on the page to run the challenge.

Running a Challenge

The running the challenge page shows the title of the challenge at the top. It also shows the distance you have traveled, the current temperature, and a timer that is keeping track of your time. There is also a start/stop button on the page to start and stop the challenge. As you go through the tasks of the challenge you will be updated on your progress.

Nearby Parks

This page starts with a map of your immediate area. Where you are is indicated with a blue marker, and nearby parks have a red pin on their location. The map can be zoomed in and moved by using standard pinch-touch motions. Pressing the white crosshairs button in the upper right corner will center the view on your current location. Pressing the "Parks List" button above will display the list of nearby parks, and these can be selected to get to the Park page. If you do not have an internet connection, these parks will not display. After pressing a location, in the bottom right corner will appear two buttons for opening Google Maps and for getting directions through Google Maps.

Parks

The parks page shows the name of the park with a list of challenges associated with that park. By clicking on a challenge you can go to the associated challenge page.

There is also an option on the page to create a challenge for the park which takes you to the challenge creation page with all the associated park information filled in for the challenge.

Friends

In the friends section you will see a list of all your friends on the application. From this list you can select a friend and go to their profile page. On their profile page you can see their points and their completed challenges.

Search Parks

In search parks you can type in a park name or a location and it will pull up a list of parks with a similar name or near the location you typed. When you click on one of the options you will travel to the park page for the selected park.

Challenge Creation

This page is used for making your own challenges to complete. To start, fill out the fields for the name of your challenge, and give a good description. Then use the Change Park button to select a park the is in you near vicinity. It is expected that you will be at the park when creating these challenges, so only nearby parks will appear in the list. Once this is complete, click the "Next" button at the bottom. On this page is a list of tasks for the user to complete. A challenge needs to have at least one task, so one is already in the list. More tasks can be added by pressing the "Add Objective" button. For each task there are several different parameter that can be filled out.

Distance: This can be given in several different units. A minimum distance means the user will have to run at least that far before the task is completed. A maximum distance means that the user can only go up to that far for the duration of the task. This is useful to make sure a user stays within certain bounds, or must take the path less traveled.

Time: Whether seconds, minutes, or hours, time is invaluable in the design of challenges. Adding a minimum time forces the user to wait for an elapsed time,

useful when combined with another task parameter. A maximum time instills a sense of urgency, turning the task into more of a race.

Temperature: This parameter really only needs to be assigned to one task to be effective. Minimum temperature means the challenge must be performed in weather warmer than the designated temperature, while maximum temperature is just the opposite. Used to create seasonal challenges.

Location: There are two different locations which can be assigned to a task. The start location means that the user must go to that location before the task can even be started. This means that if, for example, the task had a minimum time, the task would not begin counting towards the time until the user had reached the start location. The end location is similar, but for ending the task. The task will not be marked complete until the location has been reached after the rest of the task's parameters have been met. Although there are only two locations assigned per task, multiple tasks can be chained to create a track of sorts, with each location being a checkpoint.

Web Application

Getting Started

To access the website go to <http://may15.icmonroe.com/>.

Creating an Account

Once you are on the website, if you don't have an account you will need to create one. Click on Sign Up in the top right corner to begin the registration process. Complete the registration form and you will be redirected back to the starting page and now you will be able to log in.

Logging on

On the main page, you will need to click on the Log In button in the top right corner to begin the log-in process. Enter your username and password that you created

when you created your account and click the Log In button below the form. If you entered your information correctly, you will be directed to the user homepage.

Navigating the User Home Page

From the user home page you can see your current statistics, which include your total points, the highest points you've received from a challenge, that last points you've earned, and the number of challenges you have completed. Below your profile picture and other information, you will also see a list of all of the challenges that you have completed. You can click on these challenges to be taken to a description page for that challenge. At the top of the page you will also notice a navigation bar.

Navigation Bar

At the top of every page is the navigation bar. This navigation bar contains links to the Explore parks page, the Create challenges page, as well as a link back to the home page.

Explore Parks

If you navigate to the explore parks page, you will be taken to a page that will display a map with markers pointing out all of the parks near to your current location. If you click on a marker you will be shown the name of the park as well as a link you can click to be taken to the information page for that park.

Park Information Page

The park information page contains a list of all of the challenges associated with a park. If you click on one of the challenge items, you will be redirected to the challenge information page.

Challenge Information Page

The challenge information page displays all of the information that is associated with the selected challenge. The information included is the name of the challenge, the park it is located in, as well as a description of the challenge.

Create Challenge

If you navigate to the create challenge page, you will be taken to another page that contains a map in it. The map will be loaded with a marker that contains your current location. You can move the marker around the map and the latitude and longitude fields below it will be automatically updated. If you click the button Search Parks, a list of parks will populate to the left of the map. The parks in the list are chosen based on distance from where the marker was last placed. Once you have selected a park, the park name will populate a form field and the map will center and zoom in on the location of the park.

Now that a park has been selected, you will be able to add tasks for the challenge. Clicking the add task button will add a new task icon to a list located to the right of the map and display a form field at the bottom of the page. It will also add two markers to the map inside of the park. These markers are the start location(green) and end location(red) of the currently selected task. There are also distance, time, and temperature fields to be filled out as well. If you create more than one task, only the currently selected task's information will be displayed in the form and only its markers will be displayed on the map. Once you are satisfied with the challenge you have created, you can press finish and the task will be saved to the server.

Appendix II

Old Interface Design

Home

Nearby Parks

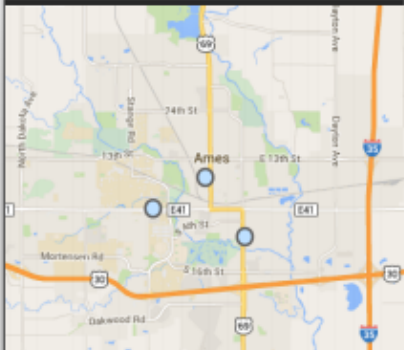
Profile

Friends

Search Parks

Logout

Nearby Parks




Park 1 .5 Miles

Park 2 1.25 Miles

Park 3 2.5 Miles

Profile



Parks Visited: 7

Challenges Completed: 20

Total Score: 330 pts

Rank: 12

Friends

John Doe 250 pts

Jane Doe 115 pts

Abe Lincoln 315 pts

Tyler Durden 780 pts

Ron Swanson 2050 pts

Dwight Schrute 560 pts

Andy Dwyer 430pts



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

Old User Interface Description

Mobile application flow

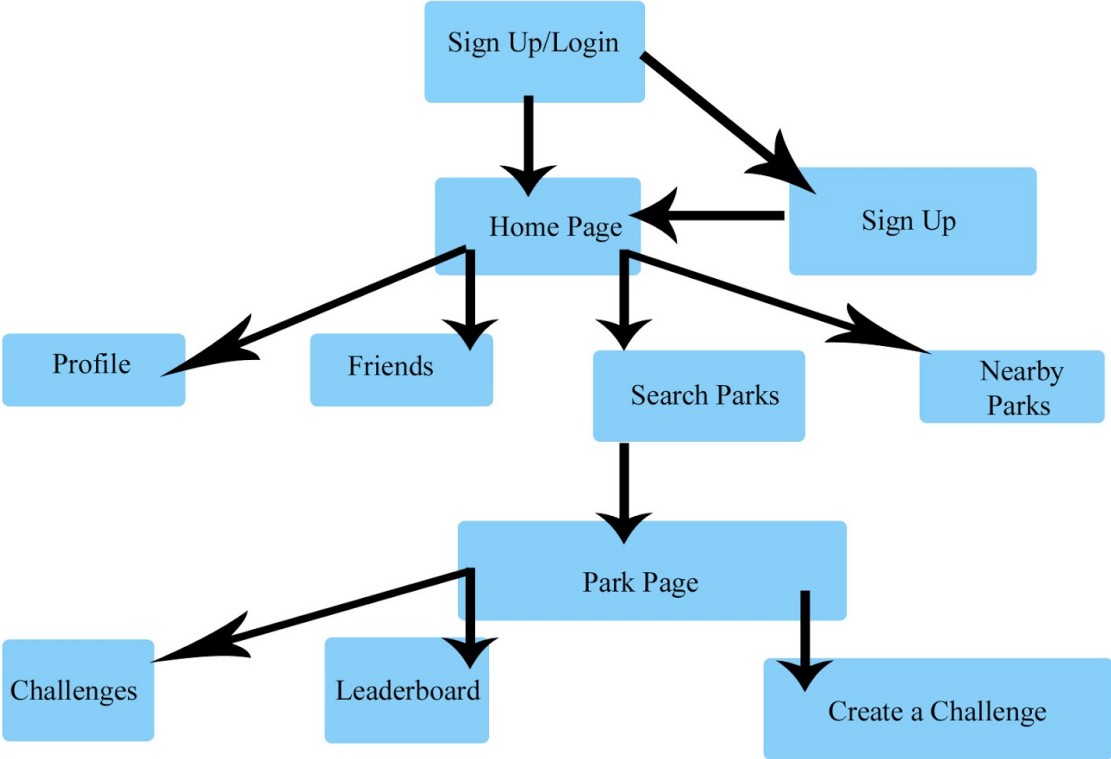


Figure 11: Page Flow Diagram