Location-aware Application to Enable Improved Physical Fitness
"Walk Wars"

Project Team 7:

Chun Sing Tsui

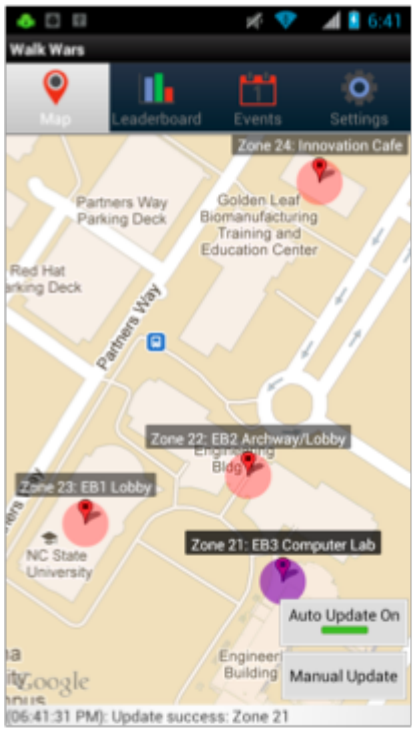
John Manby

Josh Crissman

Michael Oglesby

Table of Contents

I. Executive Summary



The purpose of this project was to build a social mobile application that utilizes Wi-Fi localization to track user movement/distance and, thus, promotes physical activity/fitness. Our sponsor was Dr. Michael Devetsikiotis, Director of the Network Performance Research Group at N.C. State University. Requirements included developing a mobile application for the Android platform that tracks user movement between pre-specified locations via Wi-Fi localization, and keeps track of a “rough” distance that has been walked by the user. We chose a zone-based localization scheme (localizing to area-specified zones, e.g. a computer lab), defined four zones, and decided to title our application, “Walk Wars,” to emphasize the physical fitness and social aspects/requirements. Our application (see screenshot above) features “Map,” “Leaderboard,” “Event,” and “Settings” screens to enable gameplay, and relies on a web server backend featuring PHP scripts and a MySQL database for data handling. In the end, despite a few major roadblocks along the way, all requirements were met.

II. Project Report

A. Introduction

The purpose of our project is to develop an application for mobile devices to track the location of users through WiFi-based localization. The application will display the location of a user on a map, in addition to presenting a leaderboard of the distance each user has traveled. This will be used in a game we are calling Walk Wars, where participants will use the app to compete who can walk the farthest distance on Centennial Campus. Walk Wars, in addition to being a social tool, could potentially be a useful way to improve physical fitness for students or other users.

The sponsor of our project is Dr. Michael Devetsikiotis, Director of the Network Performance Research Group. One of our key contacts is Andrew Williams, who is a previous graduate student that was in charge of developing the current implementation of the WiFi localization system. Our team members are Chun Sing Tsui, John Manby, Josh Crissman, and Michael Oglesby.

B. Background

The project idea is an extension of a research project by the NPRG group that utilizes indoor WiFi-signal strengths to determine location. The previous application of this localization technology was a Risk type game with rooms in EB2 as different conquerable areas. For this design project, Dr. Devetsikiotis wants to utilize data sets collected from participants in future research while expanding the application of this localization technology to promote physical fitness with a social aspect tied in. Overall, this project will improve upon the accuracy of the localization, as well as provide a new application of this technology.

C. Product Requirements

Mobile Application

(R-1) The application shall be compatible with the Android version 2.2 or newer.

(R-2) The application shall track the movement of the user between pre-specified locations using a Wi-Fi-based localization system, as described in this document.

(R-4) The application shall keep track of the distance (sometimes referred to as “points” or “score” in this document) that has been walked by the user and the user’s team.

(R-5) The application shall rely on a server backend for handling of data.

User Interface

(R-6) The user shall initially be presented with a screen that shows his or her location on a map of centennial campus.

(R-7) The user shall be able to access links to a leaderboard screen, an events screen, and a settings screen.

(R-8) The leaderboard screen shall present the user with a list of the teams that have collectively walked the greatest distance, and show the distance walked by the user's team.

(R-9) The event screen shall provide information regarding a bonus event, which shall be randomly generated daily.

(R-10) The settings screen shall present the user with the option to create a new user, create a new team, join a team if he/she has not already done so, and log in as a different user.

Server Backend

(R-11) The application shall utilize a LAMP stack backend, consisting of PHP scripts and MySQL database, for data handling.

(R-12) The backend will be hosted on an NCSU personal locker.

D. Design Alternatives

Heat Map

One possible design alternative is using a heat map in order to calculate the location of a user rather than using the pre-specified "zones." This method requires making a map of the often oblong and occasionally discontinuous shape of the coverage of each access point noting the areas of strength and weakness. While this method can guarantee a large amount of accuracy, the difficulty of code implementation and the high number of measurements required prove this method to be infeasible within the time required for this project.

Signal Strength Matching

Another method could be to take measurements on a multitude of points in the area that the application is covering and and store them on a server backend, and determining a location based on the closest matching point taken from the application. This method proves to be inaccurate due to the constantly changing strengths of access points, even within a given location. In order to deal with these constantly changing values, we decided on a zone-based location. If the strongest Wi-Fi signal that the user's device receives is from any one of a set of access points corresponding to a zone, the user is said to be in that zone.

iOS
Given our group's meager knowledge of Objective-C and the lack of an ability to use Dr. Devetsikiotis' group's developer licence on our home computers, we decided to use Android instead of iOS, once again due to the time constraint.

Different Aspects of Physical Fitness
Of the many aspects of physical fitness, we chose to tie walking to our application since it is something that nearly everyone is able to do, and also requires a user's location in order to determine distance, thus incorporating the Wi-Fi localization requirement. It can also be used as a game in the way that the team with the longest distance walked can be tracked.

Expansion Area
We chose our zones on centennial campus - the EB3 computer lab, the EB2 lobby/archway, the EB1 lobby, and the Innovation Cafe - as our expansion area(s) because these are four of the most frequently-visited locations on centennial campus.

E. Design Specifications

Server
The server will be an NCSU-issued personal locker. NCSU-issued personal lockers are web servers that support PHP scripts and MySQL databases. The server will feature several PHP scripts that integrate wtih a MySQL database to perform on-demand actions for the mobile application. The mobile application will "call" these PHP scripts using the exchange formats detailed in the "Server/App Data Exchange Interface" section.

MySQL Database
The server will feature a MySQL database with the following tables, with columns configured in the following manner:

<u>players</u>
```
user_id INT(11) NOT NULL AUTO_INCREMENT,
user_name VARCHAR(30) NOT NULL,
team_name VARCHAR(30) NOT NULL,
previous_zone_id INT(11) NOT NULL,
previous_zone_datetime VARCHAR(30) NOT NULL,
total_distance INT(30) NOT NULL,
PRIMARY KEY (user_id)
```

zones
entry_id INT(11) NOT NULL AUTO_INCREMENT,
zone_id INT(11) NOT NULL,
zone_description VARCHAR(30),
mac_address VARCHAR(30) NOT NULL,
lat VARCHAR(30) NOT NULL,
lon VARCHAR(30) NOT NULL,
PRIMARY KEY (entry_id)

distances
distance_id INT(11) NOT NULL AUTO_INCREMENT,
description VARCHAR(30) NOT NULL,
distance INT(11) NOT NULL,
PRIMARY KEY (distance_id)

teams
team_id INT(11) NOT NULL AUTO_INCREMENT,
team_name VARCHAR(30) NOT NULL,
total_distance INT(30) NOT NULL,
event_completed_date VARCHAR(30) NOT NULL,
PRIMARY KEY (team_id)

events
event_id INT(11) NOT NULL AUTO_INCREMENT,
date VARCHAR(30) NOT NULL,
event_zone INT(11) NOT NULL,
num_teammates INT(11) NOT NULL,
bonus_score INT(30) NOT NULL,
PRIMARY KEY (event_id)

messages
event_id INT(11) NOT NULL AUTO_INCREMENT,
user_id INT(11) NOT NULL,
timestamp VARCHAR(30) NOT NULL,
PRIMARY KEY (event_id)

activity_log
event_id INT(11) NOT NULL AUTO_INCREMENT,
user_id INT(11) NOT NULL,
zone_id INT(11) NOT NULL,
timestamp VARCHAR(30) NOT NULL,
PRIMARY KEY (event_id)

PHP Scripts
The server will feature the following PHP scripts:

create_team.php
Used for adding a team to the MySQL database (teams table); also includes functionality for optionally adding a user to the newly created team. Takes ‘user_name,’ ‘team_name,’ and ‘join’ as part of a JSON array as input, and returns a JSON array containing ‘success’ and ‘error’ as output.

create_user.php
Used for adding a user to the MySQL database (players table). Takes ‘user_name’ as part of a JSON array as input, and returns a JSON array containing ‘success’ and ‘error’ as output.

get_event.php
Used to retrieve information regarding the day’s bonus event. If no bonus event has been created for the current day, will create random bonus event (and add to the MySQL database’s events table). Takes ‘user_name’ as part of a JSON array as input, and returns a JSON array containing an array for each of the current day’s events as output. These event arrays will be named with a number, starting at ‘1,’ and will each contain the values ‘event_name,’ ‘bonus_score,’ and ‘completed.’
Note: supports returning multiple events per day to the mobile app, even though creating more than one event per day is not currently supported.

get_leaderboard.php
Used to retrieve a leaderboard featuring the team standings (all teams in the MySQL database). Does not require any input. Returns a JSON array containing an array for each of the teams as output. These event arrays will be named with a number (in order of team score), starting at ‘1,’ and will each contain the values ‘team_name’ and ‘distance.’

get_zone.php
When called, will determine the player’s current zone based on a MAC Address, update the player’s and associated team’s score (in the MySQL database’s players and teams tables) based on the player’s previous zone and current zone, create a random bonus event (in the MySQL database’s events table) if one has not already been created, and award bonus points (in the MySQL database’s players and teams tables) if the player’s team has “completed” the bonus event. Takes ‘mac_address’ and ‘user_name’ as part of a JSON array as input, and returns a JSON array containing ‘current_zone_id’ and ‘new_total_distance’ as output.

log_message.php
Used to log a message event in the MySQL database (messages table). Takes a JSON array containing ‘user_name’ as input.

login_user.php
Checks to see if a user exists in the MySQL database (players table). Takes a JSON array containing ‘user_name’ as input, and returns a JSON array containing ‘success,’ ‘team_name,’ ‘total_distance,’ and ‘error’ as output.

populate_databases.php
Populates the zones table (part of the MySQL databases) with zone data and the distances table with distance data. Does not require any input and provides no output. Should not ever be called by the mobile app; should only be run once, as part of initial setup (see “Construction Details”).

setup_databases.php
Drops and then re-creates all MySQL tables (creates empty tables). Should not ever be called by the mobile app; should only be run once, as part of initial setup (see “Construction Details”).

switch_teams.php
Used for switching a user’s team affiliation in the MySQL database (players table). Takes ‘user_name’ and ‘team_name’ as part of a JSON array as input, and returns a JSON array containing ‘success’ and ‘error’ as output.

Mobile Application
The mobile application will be developed on the Android platform using the standard Java programming language. It will communicate with the backend server to perform actions (updating user locations, fetching informations, etc.) using exchange formats detailed in the “Server/App Data Exchange Interface” section.

The app should target at least Android 2.2 (API level 8) to maximize the number of people that can use this app. Since this app utilizes the Android Google Maps API, the build target should be the level 8 “Google APIs” and not the regular “Android 2.2” target.

Mobile Application GUI
The app consists of four main tabs for navigation:
- Map
- Leaderboard
- Events
- Settings

Map
This tab will contain a mapview that shows all of the available zones that the server will recognize. It will also show the user’s location by displaying their previous and current zone if they are in range. There should also be two buttons overlaying on top to give the user options to auto-update or trigger a manual update.

Leaderboard
This tab will contain a list of teams sorted by their total score.

Events
This tab will contain a list of events that players can participate in to gain bonus points. By long clicking on a certain event item, the app will generate a premade message and utilize the Android built in messaging system to allow the user to easily share the event.

Settings
This tab will show the user's information including their user name, score and team name, as well as allow the user to perform account management actions such as user/team creation or logging in as existing user.

Server/App Data Exchange Interface
The JSON structure will be used for the server/app data exchange. See "PHP Scripts" section for JSON content details for each script.

F. Design Description

Server
See the "MySQL Database" and "PHP Scripts" sections for detailed information on the database and scripts, respectively. See the "Mobile Application" section for details on when the PHP scripts are called.

MySQL Database

players

user_id -	uniqe identifier (number)
user_name -	username
team_name -	name of user's team
previous_zone_id -	zone id number of user's previous zone
previous_zone_datetime -	date and time that user was last at previous zone
total_distance -	user's total score

zones

entry_id -	unique identifier (number)
zone_id -	number used to identify physical zone
zone_description -	string description of zone (optional)
mac_address -	wireless access point MAC address associated with zone
lat -	lattitude of zone's center point
lon -	longitude of zone's center point

distances

distance_id -	unique identifier (number)
description -	string, “XtoY” (i.e. “21to23” for distance from zone 21 to zone 23)
distance -	distance between center of zone X and center of zone Y

teams

team_id -	unique identifier (number)
team_name -	team name
total_distance -	teams’ total score
event_completed_date -	last date that the team completed a bonus event on

events

event_id -	unique identifier (number)
date -	date corresponding to event
event_zone -	zone corresponding to event
num_teammates -	number of teammates that have to be in the zone to complete the event
bonus_score -	bonus points the team receives for completing the event

messages

event_id -	unique identifier (number)
user_id -	id number of user who sent message
timestamp -	time that user sent message

activity_log

event_id -	unique identifier (number)
user_id -	id number of user
zone_id -	zone that user was in
timestamp -	time that user was in zone

PHP Scripts

KEY:
$name - PHP variable
‘name’ - MySQL database table column

create_team.php

1 Extracts $user_name (key=”user_name”), $team_name (key=”team_name”), and $join (key=”join”) from an input JSON array.
2 Checks to see if a team with ‘team_name’ == $team_name exists in the teams table. If a team exists, sets $error equal to an error message.
3 If a team doesn’t exist, adds a team with the ‘team_name’ = $team_name to the teams table (sets ‘total_distance’=0 and ‘event_completed_date’=”never”).
4 If this is successful, sets $success equal to a success message.

5 If not, sets $error equal to an error message and $join to false.
6 If $join is true, checks to see if a user with ‘user_name’ == $user_name, exists in the players table. If a user exists, switches his/her ‘team_name’ in the players table to the value of $team_name.
7 If this is successful, sets $success equal to a success message.
8 Otherwise, sets $error equal to an error message.
9 If a user doesn’t exist (from step 6), sets $error equal to an error message.
10 Populates and returns a JSON array with the values of $success (key=”success”) and $error (key=”error”).

<u>create_user.php</u>

1 Extracts $user_name (key=”user_name”) from an input JSON array.
2 Checks to see if a user with ‘user_name’ == $user_name exists in the players table. If a user exists, sets $error equal to an error message.
3 If a user doesn’t exist, adds a user with ‘user_name’ = $user_name to the players table (sets ‘team_name’=”none,” ‘total_distance’=0, ‘previous_zone_id’=0, and ‘previous_zone_datetime’=”empty”).
4 If this is successful, sets $success equal to a success message.
5 If not, sets $error equal to an error message.
6 Populates and returns a JSON array with the values of $success (key=”success”) and $error (key=”error”).

<u>get_event.php</u>

1 Extracts $user_name (key=”user_name”) from an input JSON array.
2 Initializes an empty array, $json_array (return value).
3 Gets todays date and creates $date_string in the format “Y-m-d.”
4 Checks to see if an event with ‘date’==$date_string exists in the events table. If an event does not exist, adds an event to the events table with ‘date’=$date_string, ‘event_zone’=a random integer between 21 and 24 (inclusive), ‘num_teammates’=a random integer between 2 and 5 (inclusive), and ‘bonus_score’=a random integer between 50 and 500 (inclusive).
5 Sets $i=1.
6 Retrieves the values of ‘event_zone,’ ‘num_teammates,’ and ‘bonus_score’ from the events table for the first/next event in the table with ‘date’==$date_string.
7 Finds the ‘team_name’ corresponding to the user in the players table and sets $team_name equal to this value.
8 Finds the ‘event_completed_date’ corresponding to the user’s team in the teams table.
9 If ‘event_completed_date’==$date_string, sets $completed=true.
10 Creates an $event_name with the value, “‘num_teammates’ teammates at Zone ‘event_zone.’”
11 Creates an $event_array and populates it with the values of $event_name (key=”event_name”), ‘bonus_score’ (key=”bonus_score”), and $completed (key=”completed”).
12 Adds this $event_array to $json_array at index $i.

13 If another event exists with ‘date’==$date_string, returns to step 6 and increments $i.
14 Returns $json_array as a JSON array.

<u>get_leaderboard.php</u>

1 Initializes an empty array, $json_array (return value).
2 Sets $i=1.
3 Retrieves the values of ‘team_name’ and ‘total_distance’ from the teams table for the first/next team, when teams are ordered by ‘total_distance,’ descending.
4 Creates a $team_array and populates it with the values of ‘team_name’ (key=”team_name) and ‘total_distance’ (key=”distance”).
5 Adds this $team_array to $json_array at index $i.
6 If another team exists, returns to step 3 and increments $i.
7 Returns $json_array as a JSON array.

<u>get_zone.php</u>

1 Extracts $mac_address (key = ”mac_address”) and $user_name (key = ”user_name”) from an input JSON array. Converts $mac_address to all upper case.
2 Finds the value of ‘zone_id’ where ‘mac_address’ == $mac_address in the zones table. Sets $current_zone_id equal to this value.
3 Returns a JSON array containing an error message (key = ”error”) if no entries with ‘mac_address’ == $mac_address are found.
4 Finds the ‘previous_zone_id,’ ‘total_distance,’ ‘team_name,’ and ‘user_id’ for the user with ‘user_name’ == $user_name in the players table, and sets $previous_zone_id, $previous_user_distance, $team_name, and $user_id, respectively, equal to these values.
5 Returns a JSON array containing an error message (key = ”error”) if no user is found.
6 If the user is on a team, finds the value of ‘total_distance’ corresponding to the user’s team in the teams table, and sets $previous_team_distance equal to this value.
7 Returns a JSON array containing an error message (key = ”error”) if no team is found.
8 If the user has changed zones since get_zone.php was last called (i.e. $current_zone_id != $previous_zone_id) and $previous_zone_id != 0, finds the ‘distance’ in the distances table corresponding to these two zone ids, and sets $latest_user_distance equal to this value.
9 Returns a JSON array containing an error message (key = ”error”) if no distance is found.
10 If the user has not changed zones (i.e. $previous_zone_id == $current_zone_id) or if $previous_zone_id == 0, sets $latest_user_distance = 0.
11 Sets $new_user_distance = $previous_user_distance + $latest_user_distance.
12 If the user is on a team, sets $new_team_distance = $previous_team_distance + $latest_team_distance.
15 Gets todays date and creates $date_string in the format, “Y-m-d,” and $date_time_string in the format, “Y-m-d H:i.”
16 Sets the user’s ‘previous_zone_datetime’ = $date_time_string in the players table.
17 Sets the user’s ‘previous_zone_id’ = $current_zone_id in the players table.

18 Checks to see if an event with ‘date’==$date_string exists in the events table. If an event does not exist, adds an event to the events table with ‘date’=$date_string, ‘event_zone’=a random integer between 21 and 24 (inclusive), ‘num_teammates’=a random integer between 2 and 5 (inclusive), and ‘bonus_score’=a random integer between 50 and 500 (inclusive).
19 Sets $event_completed_date = $date_string.
20 If the user is on a team, gets the ‘event_completed_date’ corresponding to the user’s team from the teams database, and sets $event_completed_date equal to this value.
21 If the user’s team has not completed today’s event (i.e. $event_completed_date != $date_string), retrieves the values of ‘event_zone,’ ‘num_teammates,’ and ‘bonus_score’ corresponding to today’s event from the events table. Sets $event_zone, $num_teammates, and $bonus_score, respectively, equal to these values.
22 If the user is in the event zone (i.e. $current_zone_id == $event_zone), checks to see if his/her teammates have been in the event zone within the past minute (i.e. checks to see if each teammate’s ‘previous_zone_id’ == $event_zone and ‘previous_zone_datetime’ == $date_time_string in the players table).
23 Sets $count = the number of players in the event zone.
24 If the requisite number of players are in the event zone (i.e. $count >= $num_teammates), sets $new_team_distance = $new_team_distance + $bonus_score.
25 Sets ‘event_completed_date’ = $date_string for the user’s team in the teams table.
26 Sets the user’s ‘total_distance’ = $new_user_distance in the players table.
27 If the user is on a team, sets the team’s ‘total_distance’ = $new_team_distance” in the teams table.
28 Logs the user’s activity in the activity_log table (i.e. creates a new entry with ‘user_id’ = $user_id, ‘zone_id’ = $current_zone_id, and ‘timestamp’ = $date_time_string).
29 Populates and returns a JSON array with the values of $current_zone_id (key=”current_zone_id”) and $new_user_distance (key=”new_total_distance”).

<u>log_message.php</u>

1 Extracts $user_name (key=”user_name”) from an input JSON array.
2 Gets todays date and creates $date_time_string in the format, “Y-m-d H:i.”
3 Gets the user’s ‘user_id’ from the players table, and sets $user_id equal to this value.
4 Logs the user’s messaging activity in the messages table (i.e. creates a new entry with ‘user_id’ = $user_id and ‘timestamp’ = $date_time_string).

<u>login_user.php</u>

1 Extracts $user_name (key=”user_name”) from an input JSON array.
2 Checks to see if the user exists in the players table. If the user exists, retrieves the user’s ‘team_name’ and ‘total_distance’ from the players table, sets $team_name and $total_distance, respectively, equal to these values, and sets $success equal to a success message.
3 If the user doesn’t exist, set $error equal to an error message.

4. Populates and returns a JSON array with the values of $success (key=”success”), $team_name (key=”team_name”), $total_distance (key=”total_distance”), and $error (key=”error”).

<ins>populate_databases.php</ins>

1. Populates the zones table with zone data (i.e. creates a new entry for each ‘mac_address,’ and saves the ‘zone_id,’ ‘zone_description,’ ‘lat,’ and ‘lon’ corresponding to that ‘mac address,’ along with the value of the ‘mac_address’ itself).
2. Populates the distances table with distance data (i.e. creates a new entry for each distance; saves a ‘description’ and ‘distance’ value for each distance).
   a. See “Distance Calculation” section for information on how distances were calculated.

<ins>setup_databases.php</ins>

1. Drops (i.e. deletes) all tables in the MySQL database.
2. Recreates all required tables (see MySQL database section for table information).

<ins>switch_teams.php</ins>

1. Extracts $user_name (key = ”user_name”) and $team_name (key = ”team_name”) from an input JSON array.
2. Checks to see if the user exists in the players database. If the user exists, checks to see if a team with ‘team_name’ == $team_name exists in the teams database.
3. If the team exists, switchs the user’s team in the players database (i.e. set ‘team_name’ = $team_name).
4. If successful, sets $success equal to a success message.
5. Otherwise, sets $error equal to an error message.
6. If the team doesn’t exist (see step 2), sets $error equal to an error message.
7. If the user doesn’t exist (see step 2), sets $error equal to an error message.
8. Populates and returns a JSON array with the values of $success (key=”success”) and $error (key=”error”).

Distance Calculation

Distances (in meters) were calculated using a spreadsheet, “distance_calc.xlsx.”

Mobile Application

On the app’s first launch, the user will be directed to the settings tab to log in or create a new user. The user needs to be logged in before the app will attempt to update the user’s location with the server. Subsequent launches after the first one will direct the user to the map tab. The application mainly responds to the user input and displays the appropriate information. All major processing in done server side.

When the app is running, the user has the option to utilize the auto-update function to automatically communicate with the server. This requires the app to be running in the foreground or minimized with no other app running on top of it. Wi-Fi needs to be enabled on the Android smartphone to collect Wi-Fi information, and there also needs to be network connection in order to communicate with the server.

All server scripts called by the app is done via a custom AsyncTask class “PostJSONDataAsyncTask” that runs a background thread to do the work. When the work is finished, the class that calls the custom class will update the appropriate UI components by overriding the “onPostExecute()” method. If no post execute update is necessary, then the AsyncTask will complete and there will be no need to override “onPostExecute()”.

The below subsections will describe each of the tabs. Refer to the PHP scripts section above to see in details what type of information is passed to the server and what is sent back when calling an individual script.

Location Update

If the user has auto-update toggled on, the app will attempt to update user’s location every 10 seconds. The manual update button is independent of the auto-update toggle. Regardless of the state of the auto-update toggle, the user can trigger a manual update immediately if they desire. The app will call the “get_zone.php” script on the server for location updates.

Map

The map displays any zone information that is returned from “get_zone.php”. It will first check for the presence of an error message from the return data of the server. If error message is absent, the map will be updated with the appropriate zone and score information.

Leaderboard

The Leaderboard tab, when pressed, will attempt update the user with the newest team leaderboard information from the server by calling “get_leaderboard.php”. The returned information will be used to update the listview.

Events

The Events tab, when pressed, will attempt update the user with the any events from the server by calling “get_event.php”. The returned information will be used to update the listview. There is also an “OnItemLongClickListener” set for each event items. When an event item is pressed and held, the app triggers a messaging intent that allows the user to send a premade message via an appropriate messaging app. Every time an event item is long pressed, “log_message.php” will be called to log the action to keep track of social behaviors.

Settings
The Settings tab displays the user information including their score and team name. It also allows the user to perform account management actions: register new user, create new team, login as existing user, or join a team. Depending on the action, one of the following scripts will be called:

- create_user.php
- create_team.php
- login_user.php
- switch_teams.php

Based on the server response, pop-up toast messages and UI components will be updated appropriately to reflect user or team changes.

G. Engineering Results

Mobile Application

(R-1) The application shall be compatible with the Android version 2.2 or newer.

Requirement met - see "Mobile Application" portion of "Design Specifications" section.

(R-2) The application shall track the movement of the user between pre-specified locations using a Wi-Fi-based localization system, as described in this document.

Requirement met - users are localized to zones. Accuracy is within a few feet of the zone boundaries (i.e. the application does not "localize" the user to the EB3 computer lab until the user is within a few feet of the lab door). Movement between zones is tracked - the distance that the user has traveled is logged.

(R-4) The application shall keep track of the distance (sometimes referred to as "points" or "score" in this document) that has been walked by the user and the user's team.

Requirement met. The user's "score" is the distance (straight-line distance between the center points of the zones that the user has visited, in meters) that the user has traveled.

(R-5) The application shall rely on a server backend for handling of data.

Requirement met - see "Design Specifications" section.

User Interface

(R-6) The user shall initially be presented with a screen that shows his or her location on a map of centennial campus.

Requirement met - see "Mobile Application" portion of "Design Specifications" section.

(R-7) The user shall be able to access links to a leaderboard screen, an events screen, and a settings screen.

Requirement met - see "Mobile Application" portion of "Design Specifications" section.

(R-8) The leaderboard screen shall present the user with a list of the teams that have collectively walked the greatest distance, and show the distance walked by the user's team.

Requirement met - see "Mobile Application" portion of "Design Specifications" section.

(R-9) The event screen shall provide information regarding a bonus event, which shall be randomly generated daily.

Requirement met - see "Mobile Application" portion of "Design Specifications" section.

(R-10) The settings screen shall present the user with the option to create a new user, create a new team, join a team if he/she has not already done so, and log in as a different user.

Requirement met - see "Mobile Application" portion of "Design Specifications" section.

Server Backend

(R-11) The application shall utilize a LAMP stack backend, consisting of PHP scripts and MySQL database, for data handling.

Requirement met - see "Server" portion of "Design Specifications" section.

(R-12) The backend will be hosted on an NCSU personal locker.

Requirement met - see "Mobile Application" portion of "Design Specifications" section.

H. Construction Details

Server
The server used here is a personal locker web space from the the College of Engineering that is available to students upon request (https://www.eos.ncsu.edu/lockers/request/). It supports standard PHP and MySQL. The personal locker server space used for this project can be found under the following path (in the NCSU afs space): /afs/eos/lockers/people/c/ctsui/www/. The equivalent web URL is http://people.engr.ncsu.edu/ctsui/. All PHP scripts are placed under the root directory of the server space. To access any of the scripts, simply append the script name to the end of the web URL.

After all PHP scripts are placed in the locker, setup_databases.php and populate_databases.php need to be run from a web browser to initialize the database.

Mobile Application
The application is built using the Eclipse IDE, and can easily be imported into another environment. The ADT plugin is the recommended method for developing Android applications in Eclipse. In order for the app to compile successfully, the environment must have the JDK as well as the Android SDK with the appropriate API (Android 2.2, Google APIs level 8) installed.

The java code is organized into packages that separates the source files based on a logically structure. The root package under the "src" folder is *"edu.ncsu.walkwars"*, with subfolders containing appropriate files.

Java Code

edu.ncsu.walkwars.activity
This package contains all the activities that contains the UI that the user sees. It also contains the MainApplication class which holds the overall state of the app, and controls the wi-fi scanning as well as the TimerTask for managing the auto-update feature.

- EventsActivity.java
- LeaderboardActivity.java
- MainApplication.java
- MainTabActivity.java
- MapviewActivity.java
- SettingsActivity.java

edu.ncsu.walkwars.adapter
These files are custom list adapter classes that are used to display the lists for the leaderboard and events.

- EventsListViewAdapter.java
- LeaderboardListViewAdapter.java

edu.ncsu.walkwars.logic
This file contains a custom AsyncTask class that handles communicating with the server.

- PostJSONDataAsyncTask.java

edu.ncsu.walkwars.model
These files are classes that represent various objects.

- Event.java
- LatLonPoint.java
- MapItemizedOverlay.java
- Player.java
- Team.java
- Zone.java
- ZoneOverlay.java

edu.ncsu.walkwars.util
This folder contains the Util class which contains many general helper functions. The ServerAddresses class contains all of the URL's of the server scripts as static strings so it can easily be found in one place.

- ServerAddresses.java
- Util.java

XML Layout Code
The user interfaces for this application are created using the following XML layout files. Each file name is preceded by the type of layout and followed by the purpose of the layout. A file name preceded by "activity" means that the XML file will determine the GUI of the particular activity. A file name preceded by "dialog" means the file will determine the look of a dialog box. A file name preceded by "list_item" will be the layout of the individual items of a list.

- activity_events.xml
- activity_leaderboard.xml
- activity_main.xml
- activity_main_tab.xml
- activity_mapview.xml
- activity_settings.xml
- dialog_create_new_team.xml
- dialog_create_new_user.xml
- dialog_join_team.xml
- dialog_login_user.xml

- list_item_layout_events.xml
- list_item_layout_leaderboard.xml

An important thing to note here is that in “activity_mapview.xml”, there are comments containing three different Google MapView API keys. Two of those (labeled “Laptop Debug API Key” and “Desktop Debug API Key”) are specific to the keystores on the machines which this app was developed. Therefore those keys will not work if the app is deployed with a debug keystore in a different environment. However, the “Release API key” is specific to the keystore that is generated when signing this application (refer to the deployment section for more information. Depending on the purpose of the app deployment (debug or signed), the appropriate API key will need to used for the Google mapview to show up correctly. For debug purposes, a new API key will need be generated if this app is ran from a different environment.

Deployment

There are several ways to deploy the app onto an Android smartphone. Eclipse can be used to easily run a debug version onto a device, but this would require the device to be attached to the development environment.

Another way to reach a larger audience is to export a signed version of an APK file that any smartphone which meets the minimum Android version 2.2 requirement can install and run. In order to sign the application for release, a keystore file is used. The keystore file for signing can be found under the “keystore” folder with the source code. If the application is exported and signed using this keystore file, the MapView API key mentioned in the above section will need to be the appropriate one which is labeled “Release API key”.

If you wish to create your own keystore for none debugging purposes, refer to Google’s documentation on registering a MapView API key to ensure Google maps will show up correctly.

Release Keystore Information

Keystore password: ncstate

Alias: NCSU
Password: ncstate

I. Costs

There are little costs involved with implementing and use of this application. As the smartphone app is currently for use on Google's Android OS, the development of the app is free. Distribution on Google Play, Android's app market will be a one time cost of $25.

As far as the server backend is concerned, it currently has no cost in its current incarnation as it is hosted from an NCSU student locker, but if the number of users and zones were to increase more space could be required. This cost depends on the amount of space needed, but would likely not exceed more than $50 a month.

Besides the server and application, the access point infrastructure is also required. For locations where there are many access points already, such as on campus, there will be no cost for the infrastructure, but in locations where there are no WiFi access points one will have to be added. These typically cost between $25-$100 per access point, however, if used for this purpose, a lower end device should suffice.   Depending on the size of the zone, more than one access point may be needed, but for a typical classroom sized space one should suffice. 120V AC power source will also be required for each access point, but an internet connection is not required, allowing for zones to be in locations like a bus or elsewhere an internet connection is not available.

J. Conclusions

Despite multiple drawbacks, such as a vague idea to begin with, a campus wide change in access points, confusion over the CentMesh system, and a missing group member, we were able to achieve our goal of making a location aware application for improve physical fitness that included social aspects, as our requirements stated. In addition to making a fully functional application, we were also able to make an app that has clear room for future expansion that can be pursued by future senior design groups or otherwise.

We had many difficulties in this project, but it was always solved with a slight direction change that used WiFi localization in a different way. Initially we were planning on making a much more accurate localization using triangulation of WiFi signals, but due to the constantly changing and unstable nature of WiFi signals, this proved to be impractical if not impossible.

In the end, Walk Wars ended up working better than we ever expected. With a little more work, this could be ported to iOS or possibly even Windows Phone 8 or even traditional OSs such as Mac, Windows, or Linux. Also, besides just being a game to track walking, it could be used as an on campus FourSquare-like program to show where friends are, even if they are on a bus. This may be an app NC State students will use in the future and, perhaps, even the users through world will use.

III. User Instructions

1. <u>Installing the app</u>
    a. App installation is simple, load the .apk for Walk Wars on to an Android device by sending an e-mail and selecting install from the Android GMail app or straight from the PC using a USB cable.
2. <u>Use of the app</u>
    a. Use of the app is similarly simple, once starting the App up, go to the settings tab and create a new user by typing in the username of your choice as well as typing in the name of a team you would like to join. In addition to joining a pre existing team, you may also create a team by typing in the team name of your choice by tapping the “Create Team” button.
    b. Once a user is created and a team is joined, the user can begin walking and accruing points. The user will be able to see their current location/last location on the Map tab, checking their team’s score on the leaderboard by tapping the Leader Board tab, and check up on Events on the Event tab. The Event tab displays the requirements to get the bonus points for a certain event as well as if your team has completed that objective. Also, by long pressing on the event, users will open up their text messaging application to have a text with the event details in the message itself to be sent to someone of their choosing as an easy way to alert team members of an event. From the Settings tab, a user can track their personal points, as well as change teams or change users. If a user changes teams, their points earned on one team will not carry over to the new team.
    c. The application runs in the background in Android, so the app does not have to be up and viewed in order to accrue points.

CD

The following are included on a CD-ROM (in separate folders) in a sleeve located inside the back cover of this documentation package:

- Mobile application source code
- Server source code
- Softcopy of this document
- All design day presentation materials
- Installable Android application apk