Location Racing

Design Manual

By

Philip Stafford

Table of Contents

1. Introduction

This design document was produced to illustrate to the reader how the Location racing application is going to work. It describes what the application is going to do and how the application is going to do it.

This document includes information about the types of activities, i.e. use cases, which the player will be able to carry out while using the application.

The document will also explain how the application will execute the tasks that are described in the use cases.

Also included in this document are System Sequence Diagrams, a Class Diagram, the database design, a Software Stack Diagram and some mock up screenshots of how the finished application may look.

2. Context Diagram

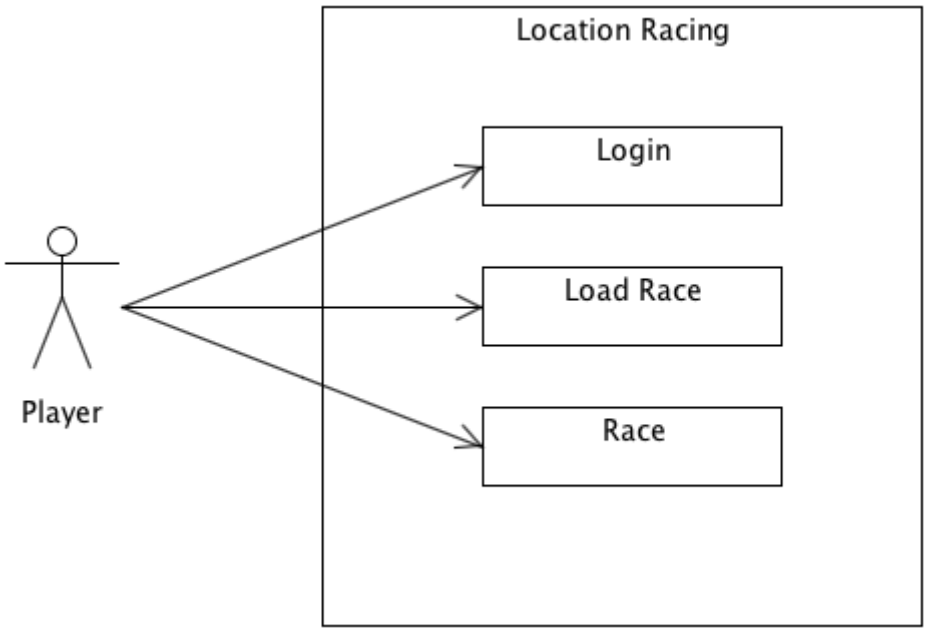


Fig 1: Context Diagram

3. Use Cases

3.1.Use Case Diagram

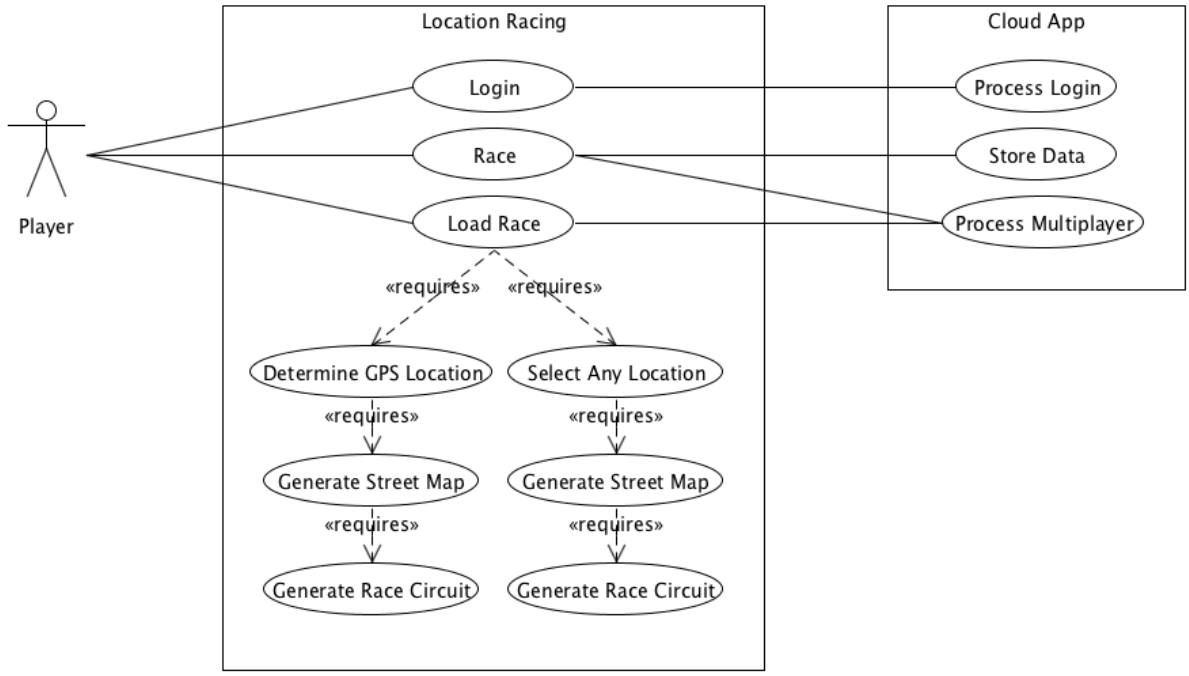


Fig 2: Use Case Diagram

3.2.Brief Use Cases

Name: **Login**

Actors: **Player**

Description: This use case begins when a player uses the racing game for the first time or if login or if the player needs to sign into the game. The player enters their login id and their password. The app communicates with the cloud application to check that the player's credentials are correct. The use case ends when the player has successfully logged in.

Name: **Load Race**

Actors: **Player**

Description: This use case begins when a player chooses to Load Race. The player chooses either the automatically detect their location or to manually select the location. Once the location has been determined the location details are shared with the Cloud app and then the Load Race use case collects the necessary info to Generate Street Map. When the street map has been generated then a circuit can be generated by the Generate Race Circuit use case. When all of the race information has been generated then the Load Race use case loads all of the track information into memory so that the race is ready to begin. It then contacts the Cloud App to see if any other players agreed to play in a multiplayer game. If so the information about the player's is loaded into memory. The Load Race use case contacts the Cloud App to see if there is any data stored in the cloud that is related to the location. This could include lap times and names of previous players.

Name: **Race**

Actors: **Player**

Description: This use case begins when a player chooses to Race. This is when the game loop is loaded.

Name: **Determine GPS Location**

Actors: **Player**

Description: This use case begins when a player chooses to determine their GPS location. The app communicates with the GPS feature to determine the exact GPS location of the player's device. Once the GPS location has been determined then the GPS coordinates are stored in the database of the mobile client. The use case ends when the GPS coordinates have been successfully determined and they have been stored in the database.

Name: **Select Any Location**

Actors: **Player**

Description: This use case begins when a player chooses to select a location to race around. The app communicates with OpenStreetMap. The player can type in any city or location they wish. OpenStreetMap searches for the location and returns results based on the players query. The player confirms a location. OpenStreetMap saves the GPS coordinates for the area and sends it to the app. The app stores this file in the Database. The use case ends when the GPS coordinates have been successfully determined and they have been stored in the database.

Name: **Generate Street Map**

Actors: **Player**

Description: This use case begins when a game is loading. The Generate Street Map function accesses the Database and retrieves the coordinates of the location that the player has decided to race around. It sends the GPS coordinates to OpenStreetMap and retrieves information from OpenStreetMap that surrounds the GPS coordinates. The information is returned to the function as a file. The file is stored within the file system. The file is parsed to determine where the roads are located. This information is then stored as a file within the file system. The use case ends when the file has been successfully parsed, converted to another file and saved within the file system.

Generate Race Circuit

Actors: **Player**

Description: This use case begins after the street map has been generated. The application retrieves the path of the file, which contains the data for the generated street map, from the database. The application calculates a route around the generated track for the cars to race around. The route is then saved as a file in the file system. The use case ends when the application successfully generates a race circuit.

3.3.Detailed Use Cases

Name: **Login**

Actors: **Player**

Main Scenario:

1. This use case begins when a player wishes to log into the application.
2. The app requests the player's user name and password.
3. The app encrypts the player's information and sends it to the cloud application.
4. The cloud application decrypts the information.
5. The cloud application validates the information and logs the player into the system.
6. The cloud application sends back a message to the application to tell it that the player is valid.

Alternative Scenario:

2. The player hasn't got an account.
3. The player registers for an account.
4. The player enters their user name and password.
5. The app encrypts the player's information and sends it to the cloud application.
6. The cloud application decrypts the information.
7. The cloud application checks to see if the user name is already in use.
8. The cloud application adds the username and password to the database.
9. The cloud application returns a message to the app to say that the registration was successful.

Alternative Scenario:

5. The information supplied by the player does not match the cloud applications database.
6. The cloud application sends a message back to the application to say that the login has failed.

Alternative Scenario:

3. The device hasn't got an internet connection.

Name: **Load Race**
Actors: **Player**

Main Scenario:

1. This use case begins when a player chooses to start a race.
2. Load Race presents the player with two options, Determine GPS Location or Select Any Location.
3. Either option will produce a location for the Load Game use case.
4. Once a location has been determined the Load Race use case sends the location to the cloud app to see if there are any other players that want to play a multiplayer game in the area.
5. Then the Load Race use case calls the Generate Street Map use case.
6. Once the street map has been generated the Load Race use case calls the Generate Race Circuit.
7. Once the race circuit is generated all of the track information is loaded into memory so that the game is ready to be played.
8. The Load Race use case contacts the cloud app to see if any players agreed to play a multiplayer game.
9. If so the player's information is loaded into memory.
10. The Race use case contacts the Cloud App to see if there is any data stored in the cloud that is related to the location. This could include lap times and names of previous players.
11. If data is found relating to the location then it is loaded into memory.

Name: **Race**
Actors: **Player**

Main Scenario:

1. This use case begins when the player is ready to race and selects the Race use case.
2. The track information is already loaded from the Load Race use case.
3. The Race use case begins the execution of the game loop.
4. The game loop begins a timer to keep track of the player's laps.
5. The game loop draws the track and the player's car to the screen.
6. The game loop applies physics to the car.
7. The game loop checks the player's car for collision detection.
8. The game loop checks for input from the player.
9. The game loop draws the new position of the car.
10. The game loop continues to run until a terminating factor is found to be true.
11. A terminating factor could be that the required number of laps has been achieved.

Alternative Scenario:

8. The game loop receives data from the cloud app regarding the multiplayer player's.
9. The game loop sends data to the cloud app regarding the multiplayer player's.

Name: **Determine GPS Location**
Actors: **Player**

Main Scenario:
1. This use case begins when a player chooses to determine their GPS location.
2. The GPS coordinates are determined.
3. The Cell ID is checked for a location.
4. The WiFi based location information is detected.
5. Determine which one is the most accurate.
6. The GPS coordinates are stored in the database with the name of the location they refer to.

Alternative Scenario:
2. The player is indoors so a GPS signal cannot be determined.

Alternative Scenario:
3. The player has no Cell signal so a Cell ID cannot be determined.

Alternative Scenario:
4. The player has no WiFi data connection so a location cannot be determined.

Alternative Scenario:
5. None of the location detection attempts were successful.
6. The GPS feature returns an error message to the app to inform it that the GPS location cannot be determined.

Name: **Select Any Location**
Actors: **Player**

Main Scenario:
1. This use case begins when a player chooses to search for a location to race around.
2. The player types in a location.
3. The app sends the location to OpenStreetMap.
4. OpenStreetMap returns a list of the results along with the GPS coordinates of each location.
5. The player chooses a location from the list.
6. The GPS coordinates are stored in the database with the name of the location they refer to.

Alternative Scenario:
4. OpenStreetMap cannot find the location the player requested.
5. OpenStreetMap returns a message to say that it cannot find the location requested by the player.

Name: **Generate Street Map**
Actors: **Player**

Main Scenario:

1. This use case begins when a game is loading.
2. The GPS coordinates of the chosen location are retrieved from the database.
3. The GPS coordinates are sent to OpenStreetMap.
4. OpenStreetMap sends back the data for the surrounding area in the form of an .osm file.
5. The .osm file is renamed and stored in a specific location in the file system.
6. The .osm file is then parsed to determine the location of all of the roads within the surrounding area of the GPS coordinates.
7. This road location information is then saved to an .xml file.
8. This .xml file is then named accordingly and saved within the file system.
9. The path of the newly generated file is stored in the database.

Name: **Generate Race Circuit**
Actors: **Player**

Main Scenario:

1. This use case begins after the street map has been generated.
2. The Generate Race Circuit use case checks the database for the path of the file which contains the information about the generated street map.
3. The application calculates a route for the player to race around.
4. A route is calculated and the details of the route are stored a file on the file system.
5. The path of the file is stored in the database and it is linked to the location.

4. System Sequence Diagram

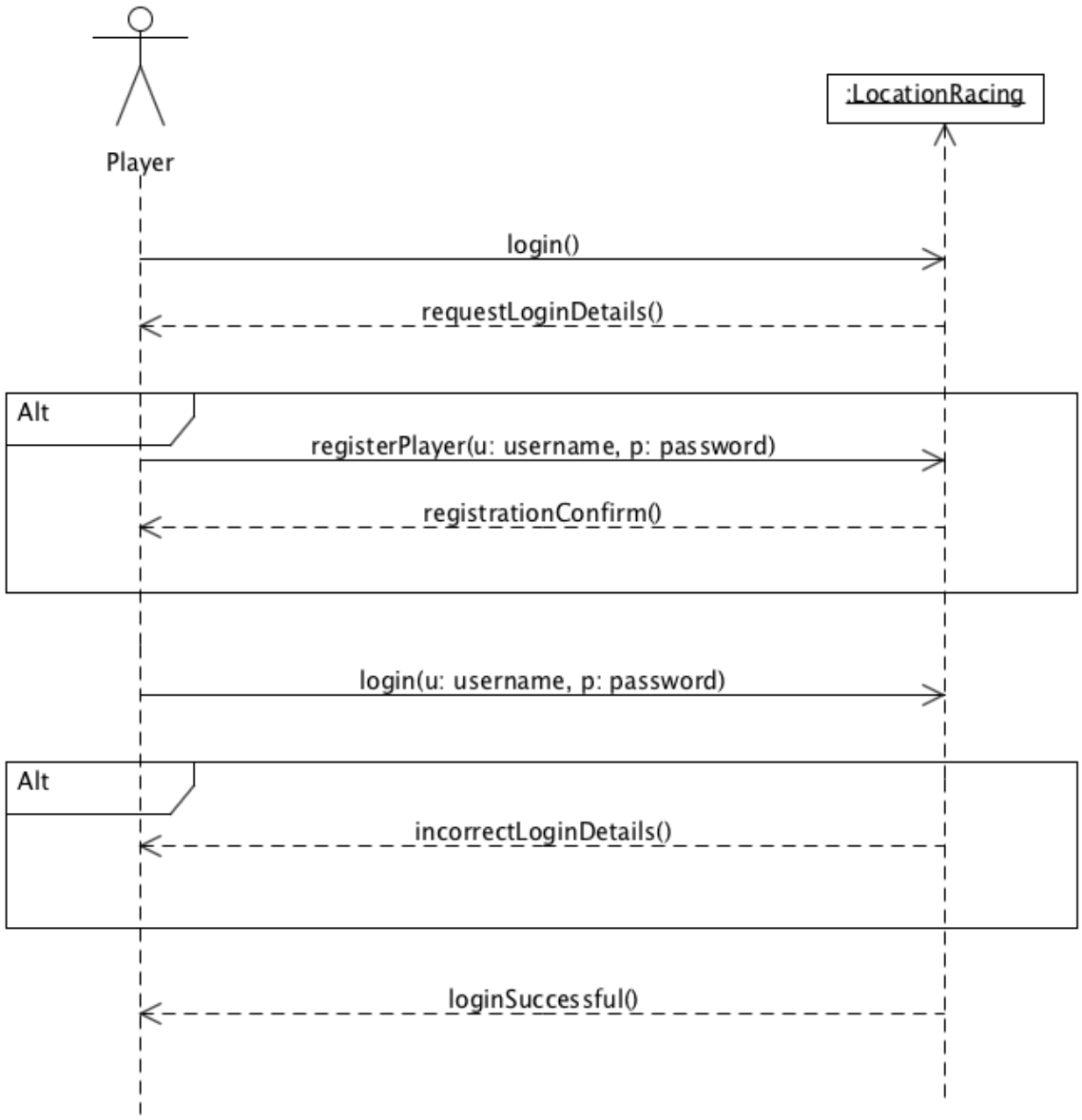


Fig 3: Login System Sequence Diagram

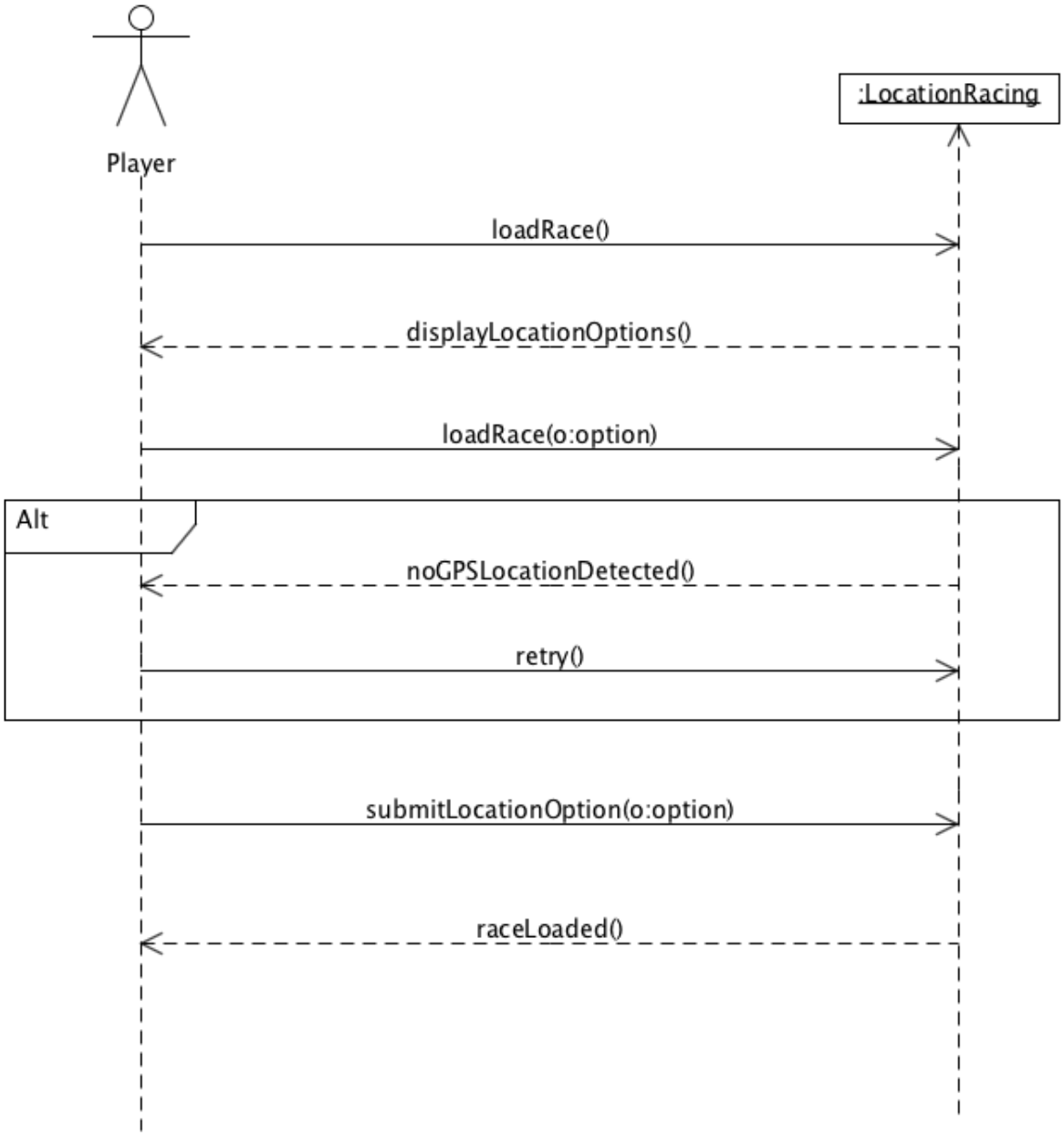


Fig 4: Load Race System Sequence Diagram

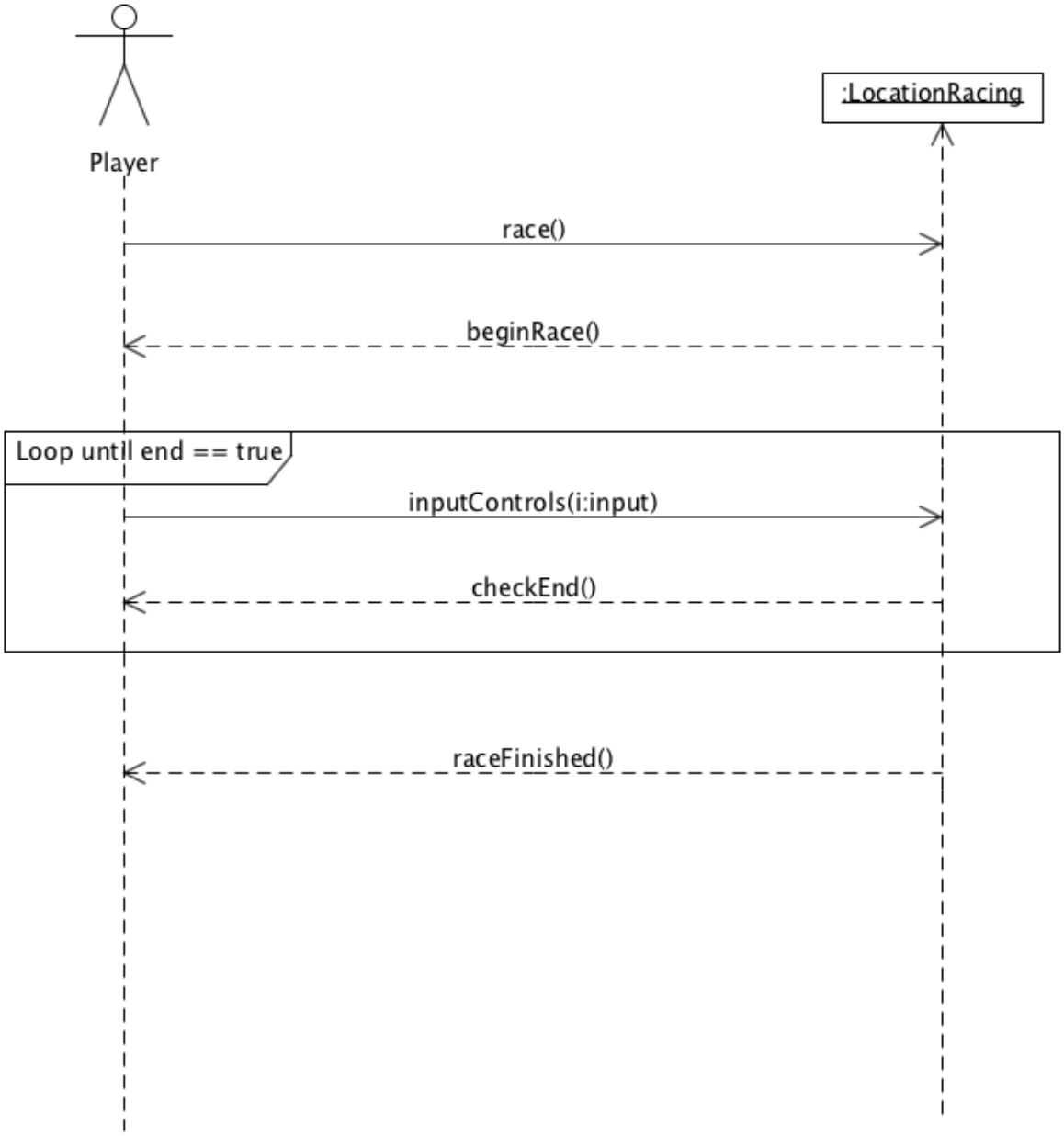


Fig 5: Race System Sequence Diagram

5. Domain Model/Class Diagram

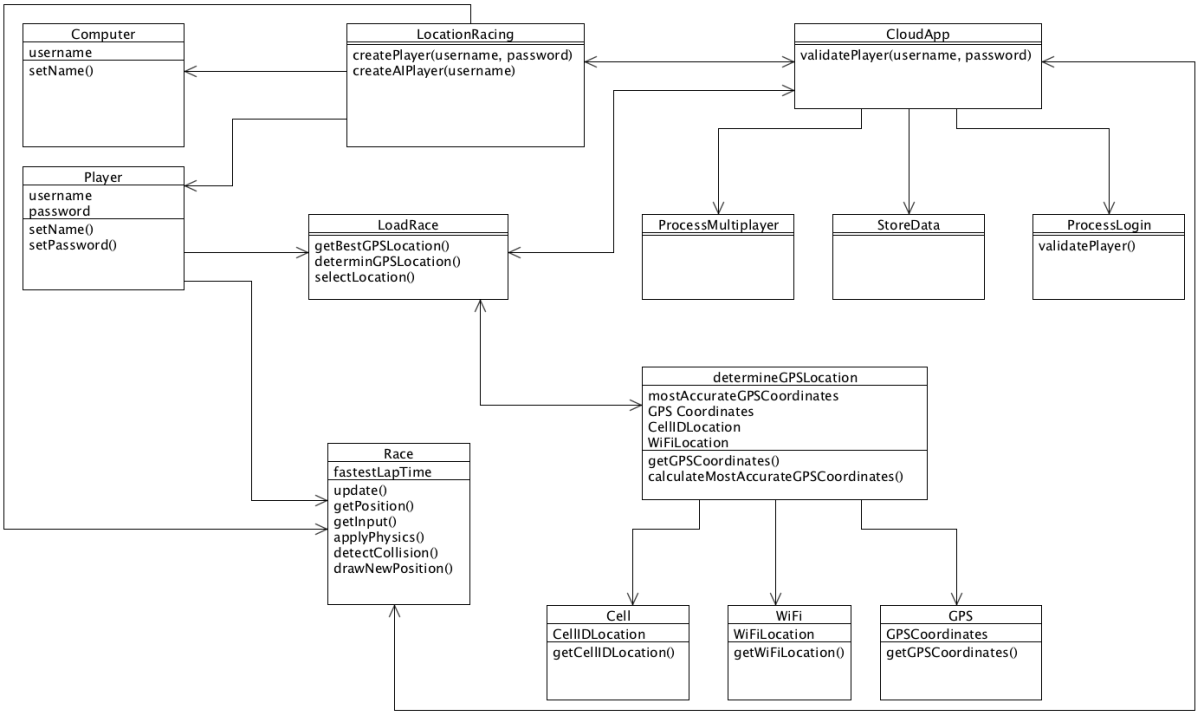


Fig 5: Class Diagram

6. Identify Database

The database tables for the Location Racing application are illustrated below.

Field Name	Data Type	
locationID	Number	The ID number of the location
locationName	Text	The name of the location
gpsCoordinates	Text	The GPS coordinates of the locaion
dateCreated	Date/Time	The date the location was created
originalLocationFilePath	Text	The path of the original file that was retrieved from OSM
modifiedLocationFilePath	Text	The path of the modified file that was created from the OSM file

Fig 6: Location Table

Field Name	Data Type	
raceID	Number	The ID number of the race
fastestLap	Date/Time	The fastest lap which was clocked in the race
dateCreated	Date/Time	The date the race was created

Fig 7: Race Table

Field Name	Data Type	
playerID	Number	The ID number of the player
playerEmail	Text	The email of the player
playerName	Text	The name of the player
playerPassword	Text	The palyers password
dateCreated	Date/Time	The date the player was created
playerType	Text	The type of player

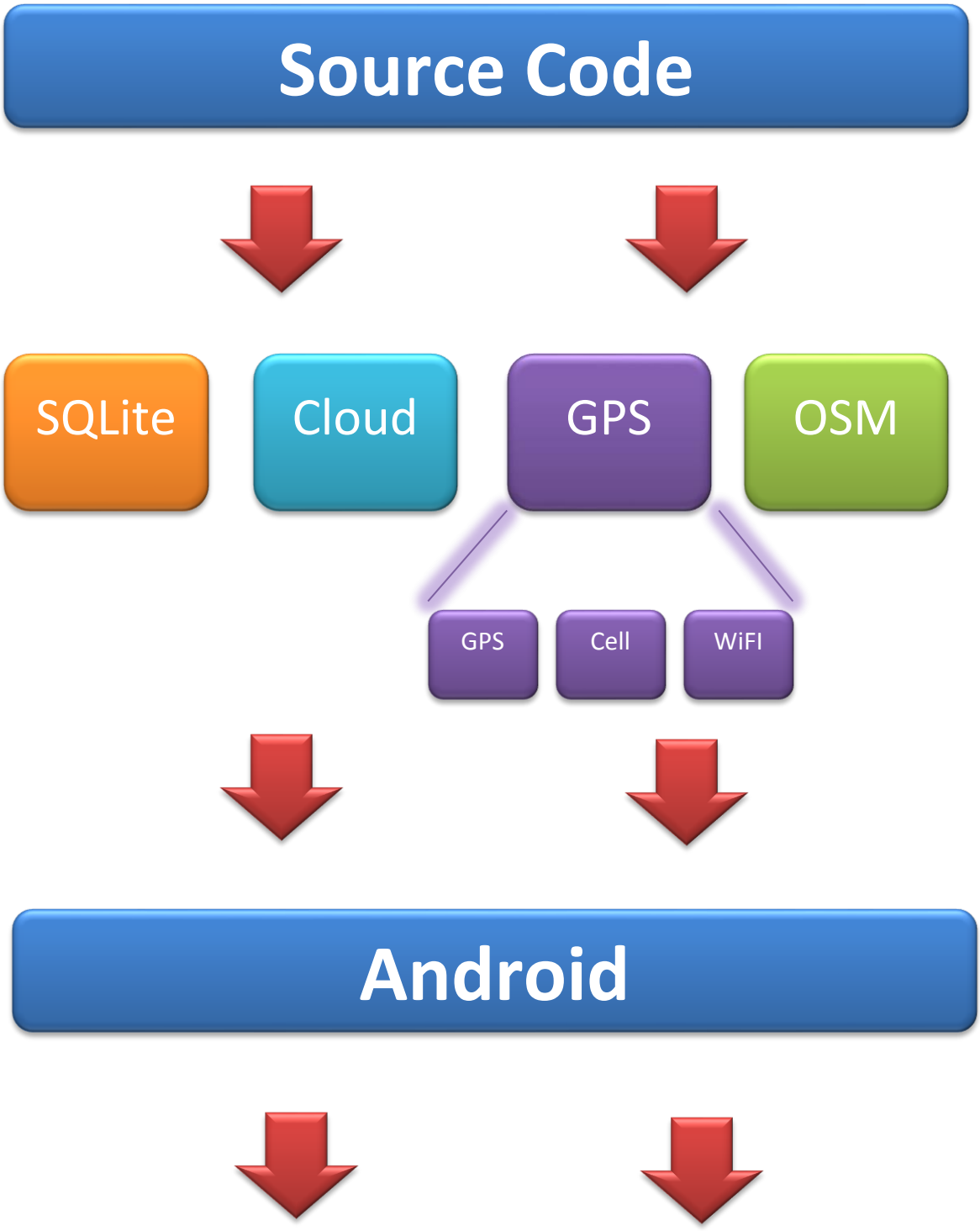
Fig 8: Player Table

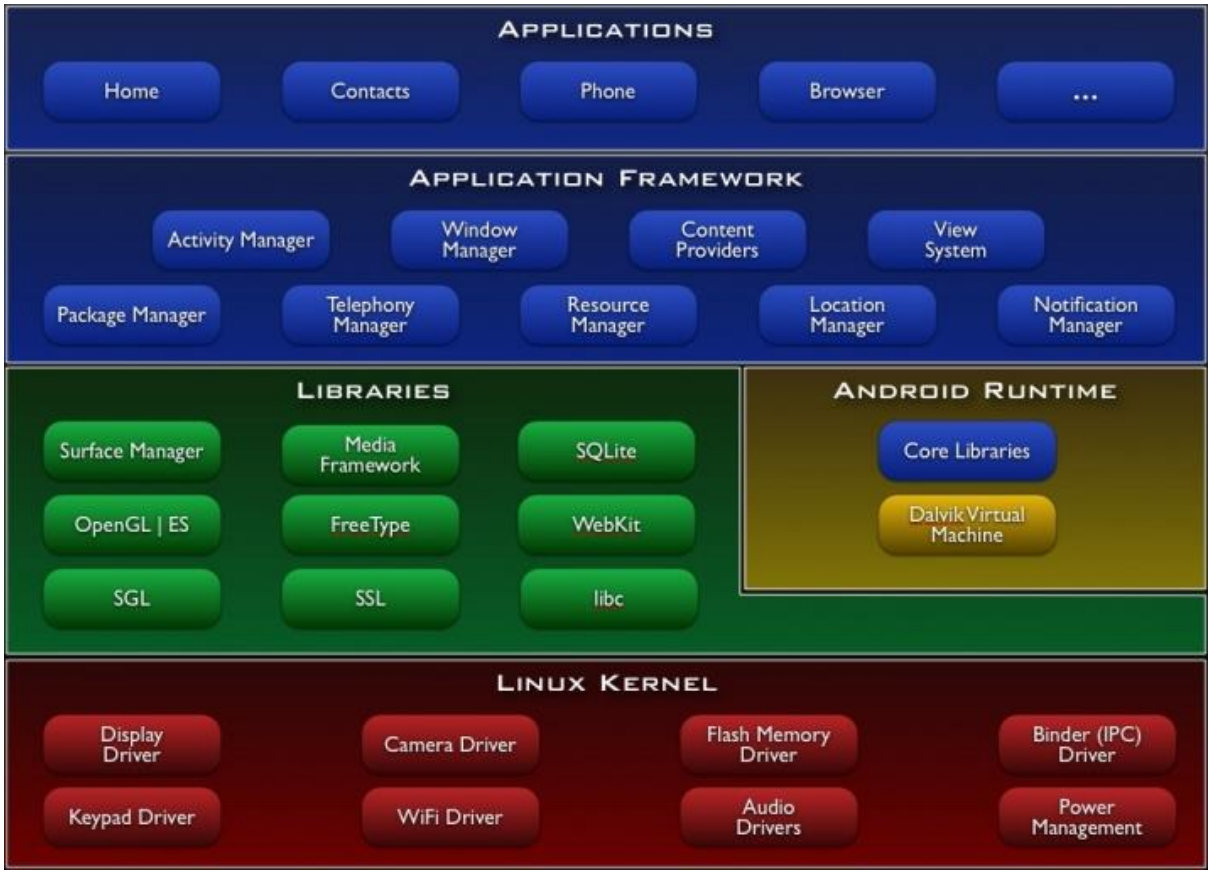
Field Name	Data Type	
aiID	Number	The ID of the player
aiName	Text	The name of the player
type	Text	The type of player
dateCreated	Date/Time	The date the player was created

Fig 9: AI Payer Table

7. Software Stack

This section illustrates the software stack and how each of the individual elements will communicate with each other.





(The University of Texas at Austin)

8. UI Mock Ups

Below are sample screenshots of what the racing game option screens and what the racing gameplay will look like.

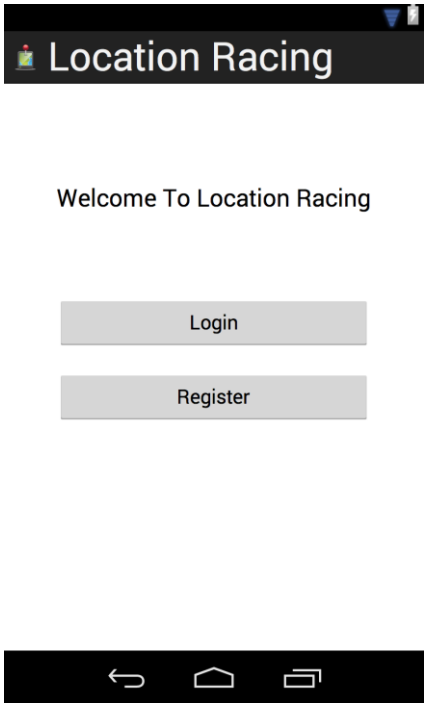


Fig 10: Welcome Screen Mock Up

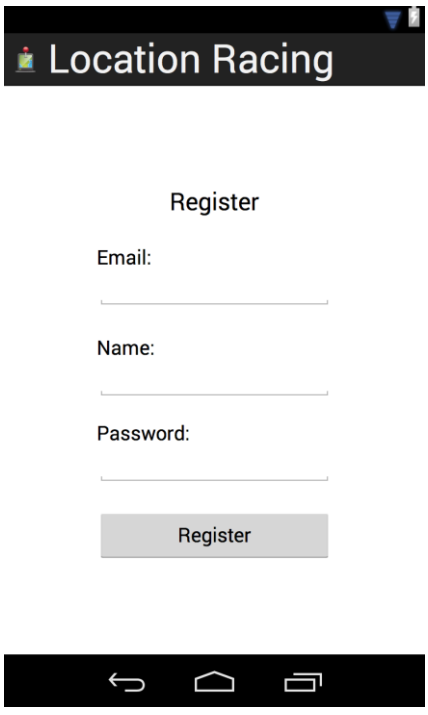


Fig 11: Register Screen Mock Up

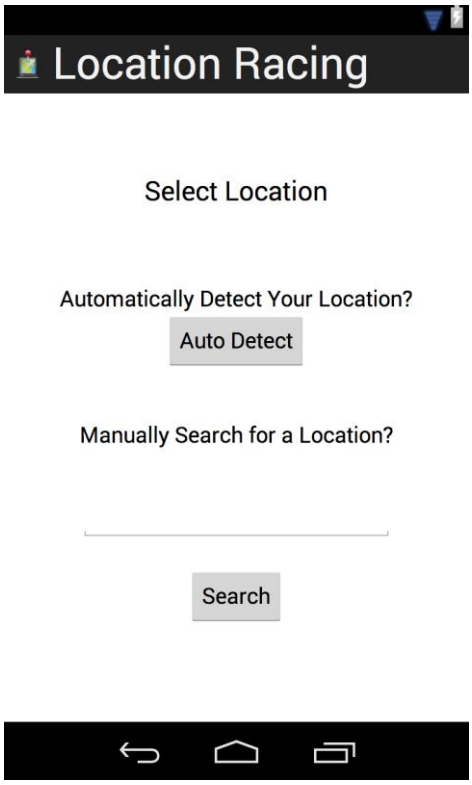


Fig 12: Location Screen Mock Up

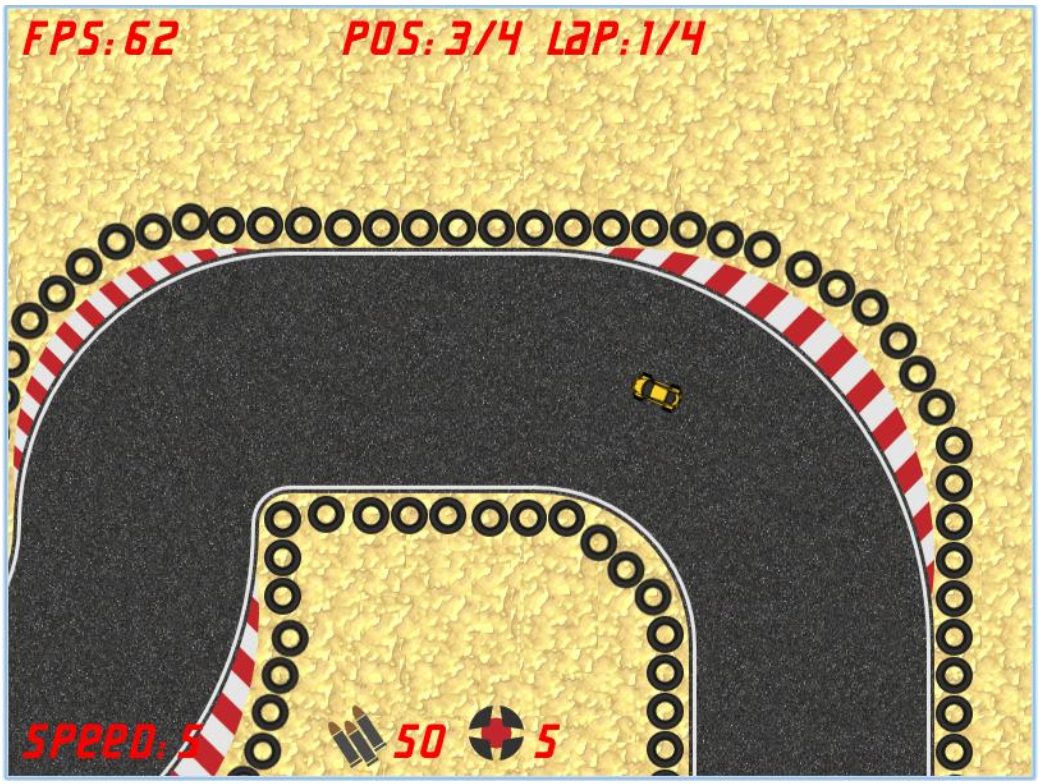


Fig 13: Track Mock Up (banditracer.eu)

9. References

Bibliography

banditracer.eu. (n.d.). *banditracer.eu*. Retrieved January 17, 2014, from banditracer.eu: http://www.banditracer.eu/

The University of Texas at Austin. (n.d.). *Android (SDK) | Open Websites*. Retrieved from The University of Texas at Austin: http://ows.edb.utexas.edu/site/collaborative-bluetooth-edumanet/android-sdk-2