**Play Boiler**

**Craig Brentz, Jake Brown, Denver Kirschling, Dylan Smith**

**Team number 7**

**Purpose Statement:**

In the world today, it’s difficult to find an effective way to meet up with peers. We will create an application that will allow peers to meet up to play pickup games together. What differentiates this app from similar systems is its specialization of sports-related events.

**Design Outline**

This application will use a client-server model to allow users to be able to meet up to play their pickup games. We have a system where the same data needs to be accessed by other people and it is easier to have it centralized in one place which means client-server best meets our needs.

Components:

The system will contain four components. The first component will be the user interface, or the client. The purpose of this component is to interact with the user and clearly and concisely deliver the requested information to the server. It will also be able to retrieve information from the server. The details of the information transferred will be covered later in the document.

The second component of the system will be a single server. The server’s purpose is to listen to requests from the client and retrieve and send the correct information back to the client.

The next component is the client-side database. Its purpose is to store information about the user on the client itself. This details of what will be stored will be described later in the document.

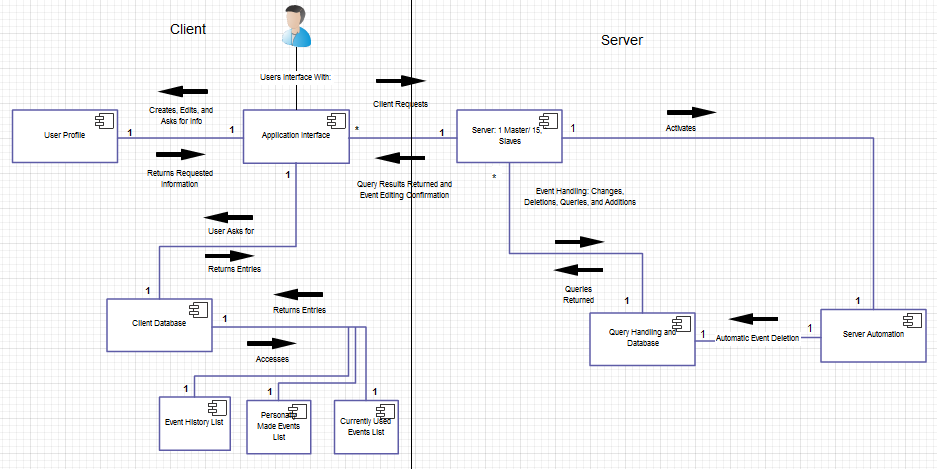
The final component is the server-side database. Its purpose is to store data on the server that can be accessed by all users. The information that will be stored wil be later described in the design details.

Interactions:

The server and server database will interact. The server will receive the information that needs to be stored and then store it into the server database.

The client and server will need to interact so that information can be passed from the server to the client. This will include information from the client that will need to be stored into the server database, and also information from the server database that the client wants to see.

Another interaction will be between the client and the client-database. The client will create information based upon user actions and then will store that information into the local database.



UML diagram showing the high-level structure of the program

**Design Issues**

Issue 1: What should the server-database do with old events that have occurred?

Option 1: The server-database should be automated to delete events that have already occurred so that the table doesn’t get too full, but the client-database will store the events that the user has been a part of.

Option 2: There should just be a flag element in the table that will say if an event has occurred or if it is going to occur.

Option 3: Have two different tables where one will store upcoming events and the other will store events that have occurred and the database will switch from upcoming to occurred.

Decision: We decided that option 1 is the design method that we are going to choose. This will stop the table from being cluttered and we don’t have a use on the server-side for events that have passed. The relevant events to the user will still be stored for the user if they ever want to view them again.

Issue 2: How should the user profile be stored on the client-side?

Option 1: The user profile should be stored in the file as JSON file that can be read by the user interface.

Option 2: The user profile should be stored as a table in the client-side database and the information can be retrieved through queries.

Decision: We decided option 2 will be the best to use. Even though the file system with a linked list would be easier to add to, the database would provide better maintainability and quicker retrieval of specific information.

Issue 3: Does there need to be a user table on the server that stores all of the user profiles from the app?

Option 1: There needs to be a table that stores all the user profiles so that a user can look up other users to see their profile

Option 2: The user profile is only for the user and shouldn’t be able to be seen by anyone else on the app.

Decision: The server needs to be able to access all of the user information so that when they join a game other users in the game can view their profile to see their specific ratings. This information will be important so that they need to know who they are playing against.

Issue 4: How to store the player list of people attending the event.

Option 1: Store the user id in an xml string in one column and parse through it to see who will be there.

Option 2: Create a separate column for each player and store a specific user id in each column.

Decision: We decided that option 1 will be the best option. It will allow the storage to be able to grow to arbitrarily large amounts without taking up a lot of extra space within the database.

Issue 5: Should the profiles have passwords associated with them?

Option 1: Do not have a password, instead have the unique hex ID of the device be how profiles are authenticated.

Option 2: Require each profile to log in with a password.

Decision: We decided on option 2, because it will be easily appended to the current table of user profiles. It will also add a level of security and allow users to carry over their profiles if they change devices

Issue 6: How should we handle security?

Option 1: Encrypt all data being sent between the client and the server in order to avoid it being

Option 2: Do not encrypt any data being sent between the client and the server.

Decision: We will choose option 2 and not encrypt any data. This is because none of the data is valuable enough to be compromised, nor will there be any incentive to retrieve another user’s data.

**Design Details**

User Interface:

User Profile Class

Will show the profile information of the user. This information will include their name, gender, age, skill levels for particular sports, sportsmanship level, bio, and alias.

Event Class

Will contain the location, type of event, time of event, who will be attending the event, and a brief summary of the event.

Search Activity Class

Will allows users to search for specific events based on their specific criteria. The criteria will include event type, event time, skill level, and sportsmanship level.

Join Class

Will allow users to join a specific event. This class will check to make sure the capacity is not full and that the user meets the requirements for the events. It will then request that the user id be stored into the server database into the correct event table entry.

Create Activity Class

Will allow users to create a form that will make a new task. This form will include event type, event location, event time, and brief summary.

Event History Classes

Will allow users to view information from a list of event classes that correspond to all the recent events that the users have participated in.

Profile Activity Class

Display a form of information from the user’s specific user profile that they can then edit and change as they see fit.

Client Database:

Event Type Table

Store a list of all the events that are available on the application. This list of events will include: Football, Soccer, Baseball, Basketball, Cricket, Disc Golf, Golf, Wallyball, Bowling, Volleyball, Sand Volleyball, Ultimate Frisbee, Ping-Pong, Floor Hockey, Dodgeball, Racquetball, Squash, Badminton, Tennis, Nerf Wars, and Other. Other will provide a space for people to insert other sports that they can think of.

Event History Table

Store 10 most recent events that the user has participated in.

User Profile Table

Will store the user id and all information from the previously mentioned user class.

Server:

Master Function

Contains the master server and listens to connections incoming from the client and assigns a slave socket to the event.

Slave Function

Accepts the incoming connections and parses through the information for the query to the database.

Query Function

Queries the database and gets the information the client requested. Then sends the information back through the save socket through the client

Change Function

Changes entries in the database as requested by the client and authenticates the identity. Sends back authentication that it happened through the slave socket.

Add Function

Adds entries into the tables in the database as requested by the client. Sends back authentication that it happened through the slave socket.

Deletion Function

Deletes entries in the tables in the database as requested by the client. Sends back authentication that it happened through the slave socket.

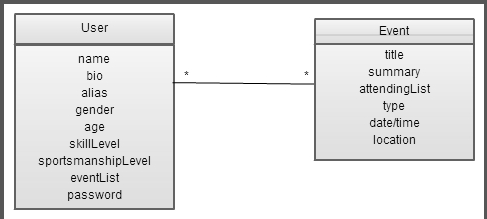
Server-Database:

User Profile Table

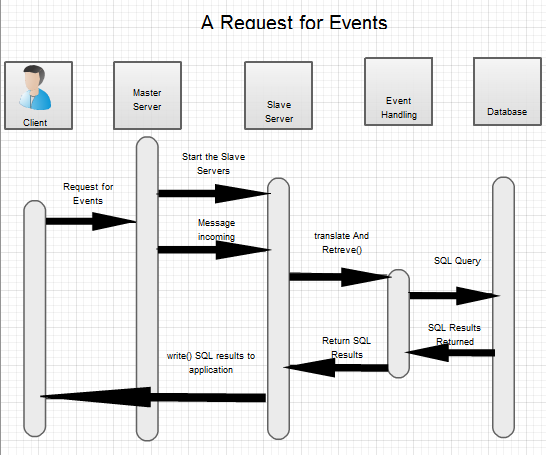
Will contain a user-id, password and all information from the profile in the user interface and store all of users’ profile’s information in the table.

Event Table

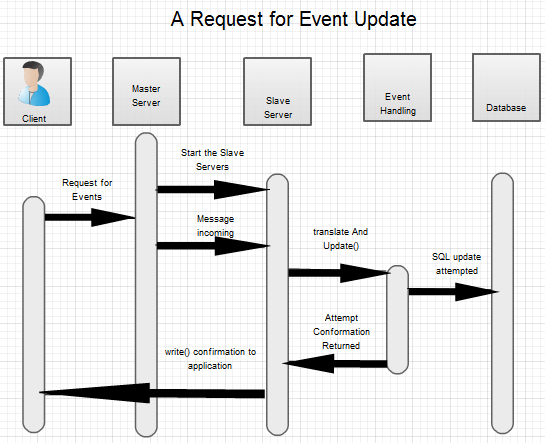
Will contain an event id and all the information an event contains. This will store all the events that the users’ have created. It will also store a list of user who say they are going to attend the event by storing them in a list in a specific column.



Class Diagram



Sequence diagram for a user requesting to view events



Sequence diagram detailing how a client updates one of their created events.