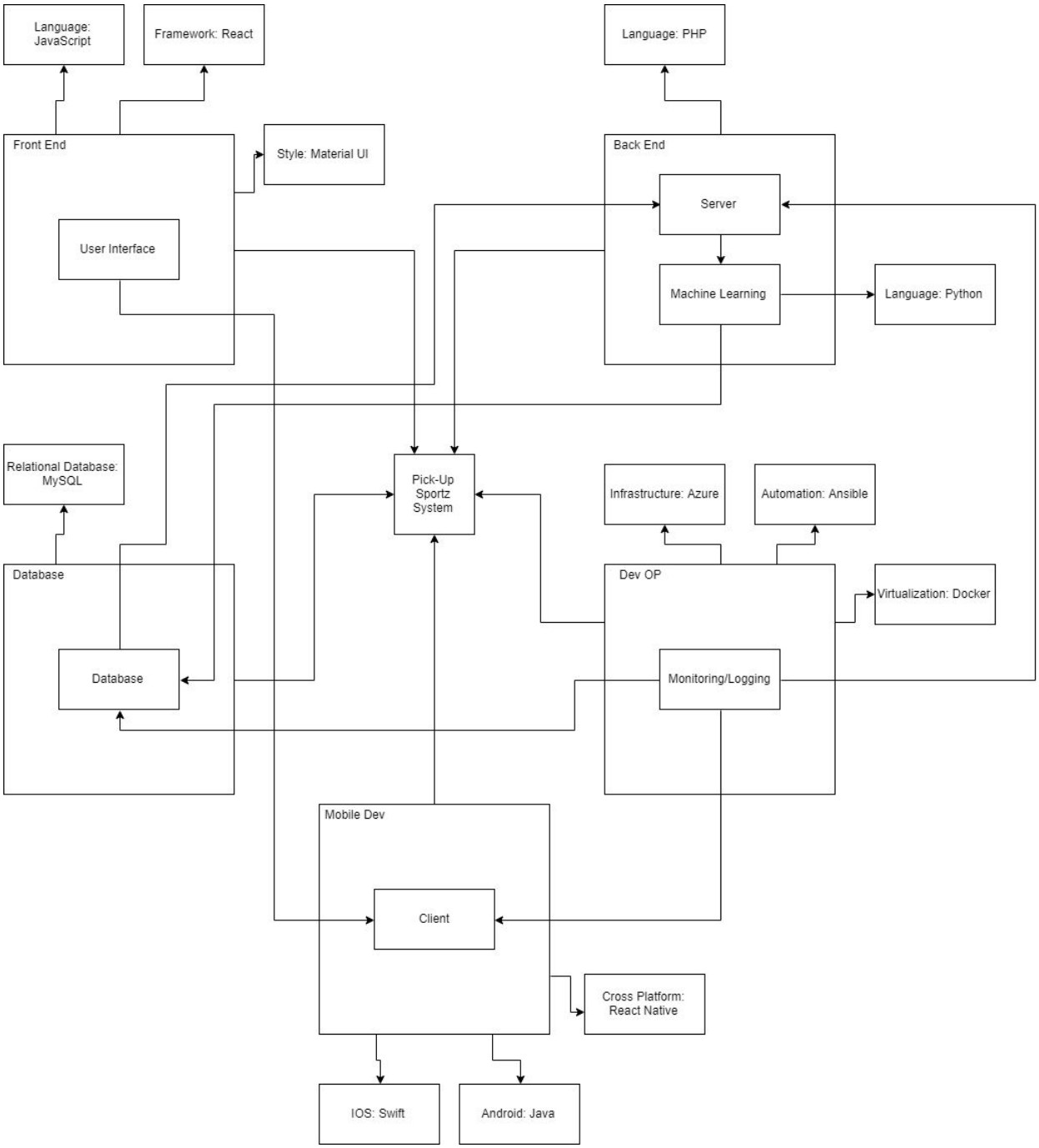
Pick-Up Sportz
Architecture and Design

By
John Him
Jamil Khan
Brandon Le
Benjamin Seo
Chaz Del Prato
Christine Duong

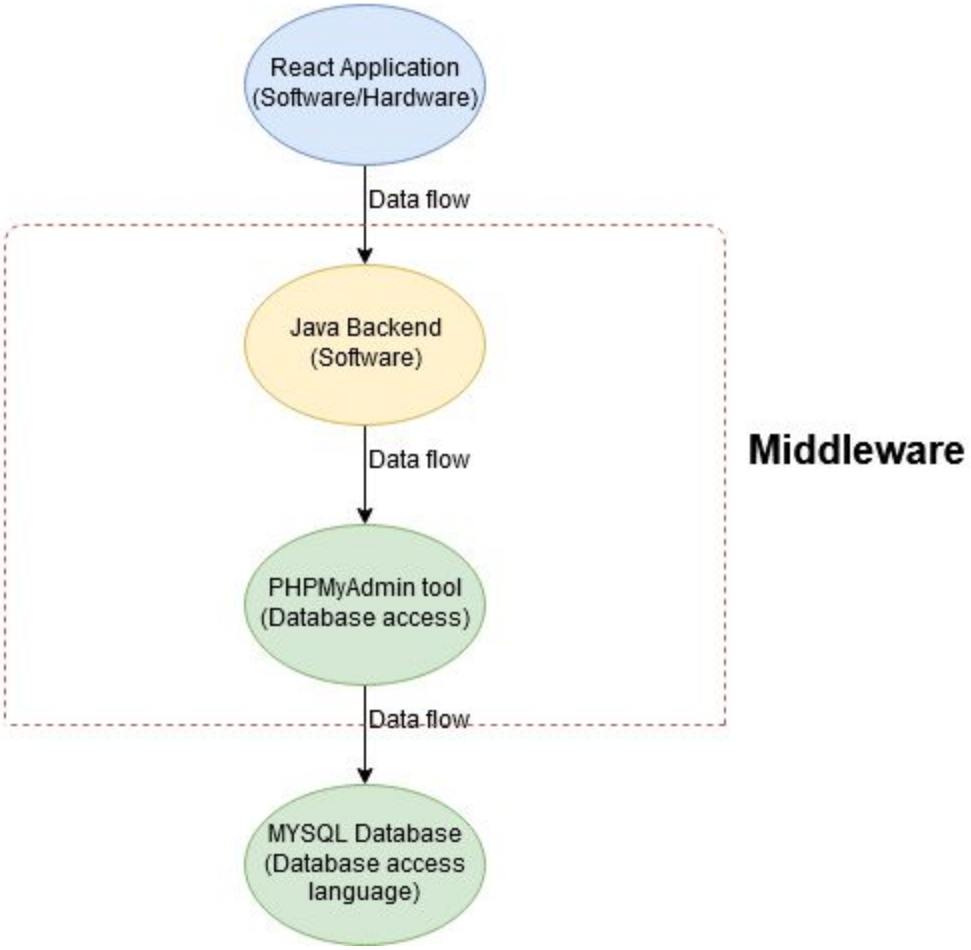
Table of Content

Diagrams

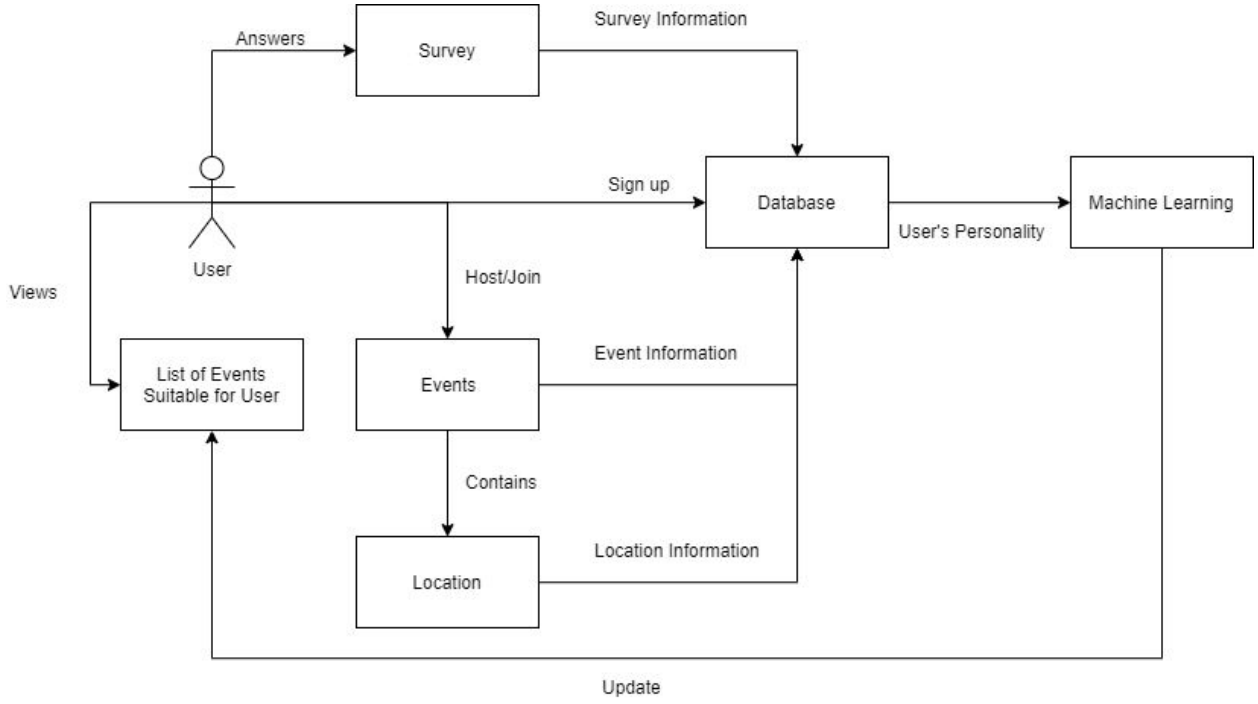
System Diagram(s) and Component Diagram(s)



Components involved in database connectivity



Information Flow Diagram



Quality and Quantity Standard

To set up the front end of our application, we will be using JavaScript through the React Native framework. It will be set up with the Material UI style. By using React Native, there is less burden on us to handle the UI interactions and makes threading a lot easier.

For mobile development, using React Native will allow us to develop the application for either the Android, iOS platform, or even both to allow the app to function across both platforms. If we choose to build the app solely on iOS then we will use Swift, and Java if we choose to make the app exclusively for Android.

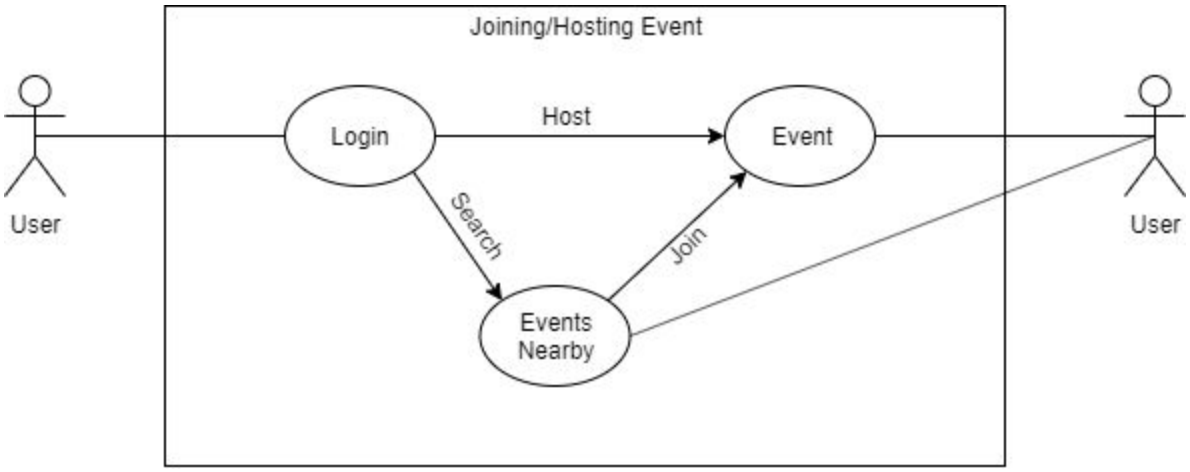
To set up the back end of our application, the language we are planning to use is PHP. This will consist of a server component and our machine learning component written in Python that will communicate with our client, database, and our DevOps.

To set up the database component, we will use MySQL as a primary foundation for our relational database. The database will obtain data from numerous components such as, client, server, monitoring/logging, and machine learning components. From this, it will work hand-in-hand with our back end.

To set up the DevOp component, it will consist of an infrastructure using Azure with an Automation, Ansible, and a virtualization, Docker. This portion will consist of our monitoring/logging components that will communicate with our client, server, and database.

Use-Case Diagrams

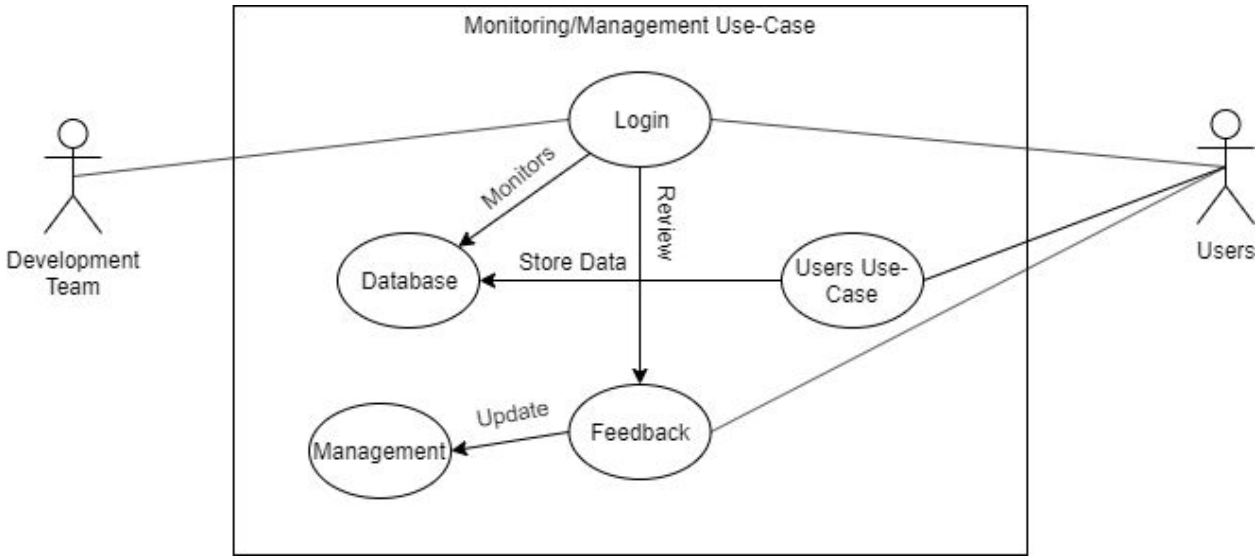
User Host/Join Use Case:



- Use-case field
  - This use case shows the action of the user when joining/hosting an event.
- Use-case name
  - Host/Join an event
- Subject area
  - Users Use Case
- Actors
  - User (left) is the actor that is hosting/joining a game
  - User (right) is the actor that is other user(s) that hosted or joined a game
- Use-case overview
  - A user (left) first logins then either search for an event nearby to join or host their own event.
  - A user (right) has either joined the event that the user (left) has hosted or joined the same event the user (left) has joined.
- Preconditions
  - User (left) must have signed up and answered the survey.
- Termination outcome
  - The use case might end when the event the user (left) joined is submitted to the system, i.e. the game has started.
  - The use case might end when the event the user (left) joined is cancelled.
- Condition affecting termination outcome
  - Cancelled event.
  - Event has passed the deadline.

    - Event has started
- Use-case description
    - Cancelled event.
        - The system will notify the users that the event has been cancelled and will close down the event.
    - Event has passed the deadline.
        - The system will notify the users that the event has passed the deadline, erasing the event from the database.
    - Event has started
        - The system will notify the users that the event has started and will save the event into the users schedule/database, while also erasing the event from the event nearby list.
- Use-case associations
    - Sign up Use Case
    - Check Schedule Use Case
    - Filter/Map Use Case
    - Look Up Other Users Profile Use Case
- Traceability to a list of other related documents, models, and products that are associated with this use case
    - Map API
    - Users Profile/History
    - Nearby Events
- Input summary
    - Event search bar/filter
    - Event descriptions/information
    - Login information
- Output summary
    - List of events nearby
    - Schedule
- Usability index (out of 10)
    - Satisfaction (7)
    - Importance (10)
    - Frequency (10)
- Use case notes
    - System must keep track of information of events, such as type of event, location, either hosted or joined, etc. so that the machine learning component can learn through the user's personality which can result in better events suited for them.

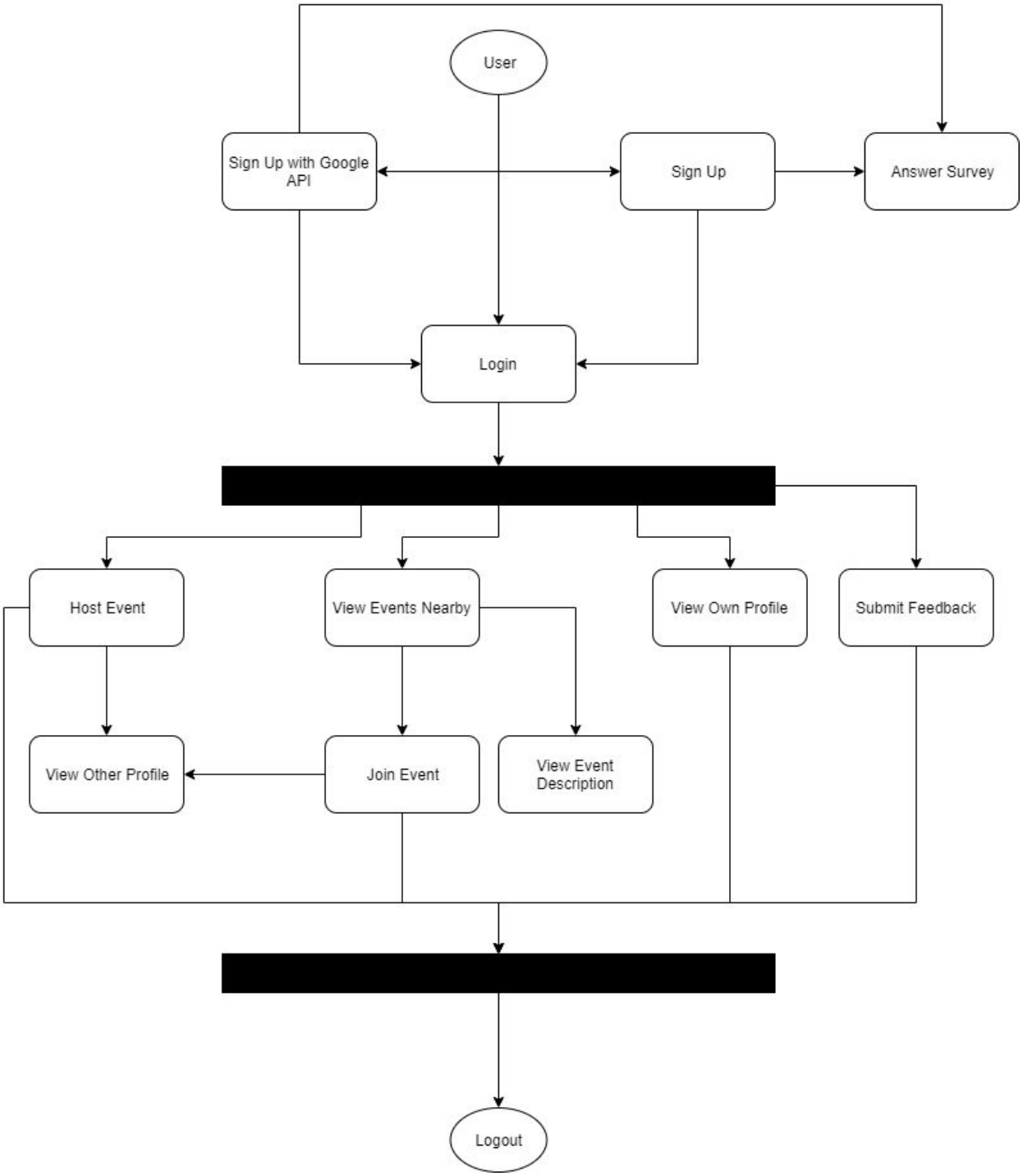
Monitoring/Management Use Case:



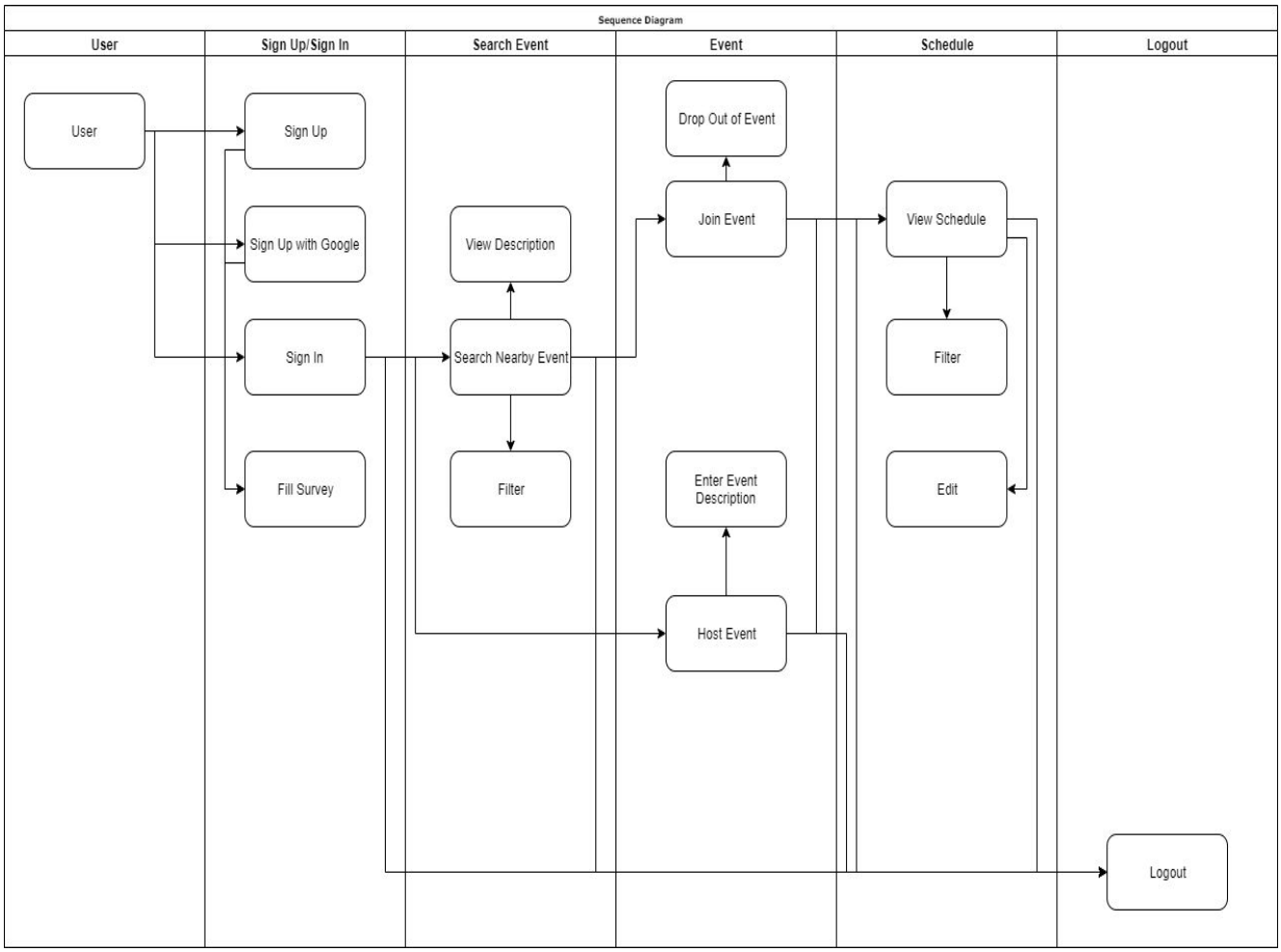
- Use Case Field
    - This use case shows what and how the development team monitors/manage in the system
- Use Case Name
    - Manage/Monitor the System
- Subject Area
    - Developers Use-Case
- Business Event
    - Login
- Actors
    - Development Team
    - Users
- Use-Case overview
    - The development team monitors the users activity; such as events joined/hosted, change in personal information, etc. while also managing feedback from users when they use the product.
- Preconditions
    - When users change their information or someway alter the database through using the application. In addition, when users submit a feedback/comment that has to do with some functionalities of the product.
- Termination Outcome
    - There is no termination outcome for monitoring and managing the application. There needs to be constant managing and monitoring.
- Condition Affecting Termination Outcome
    - No conditions
- Use Case Associations
    - User Use Case
- Input summary

    - Login information
    - Notification for change in system
    - Update Notification
    - Feedback reply
- Output summary
    - System Notification
    - Feedback reply
- Usability Index (out of 10)
    - Satisfaction (6)
    - Importance (10)
    - Frequency (10)
- Use case notes
    - This use case will allow developers to implement new functionalities according to user feedback. In addition, it will allow developers to secure the confidential information/data leaks.

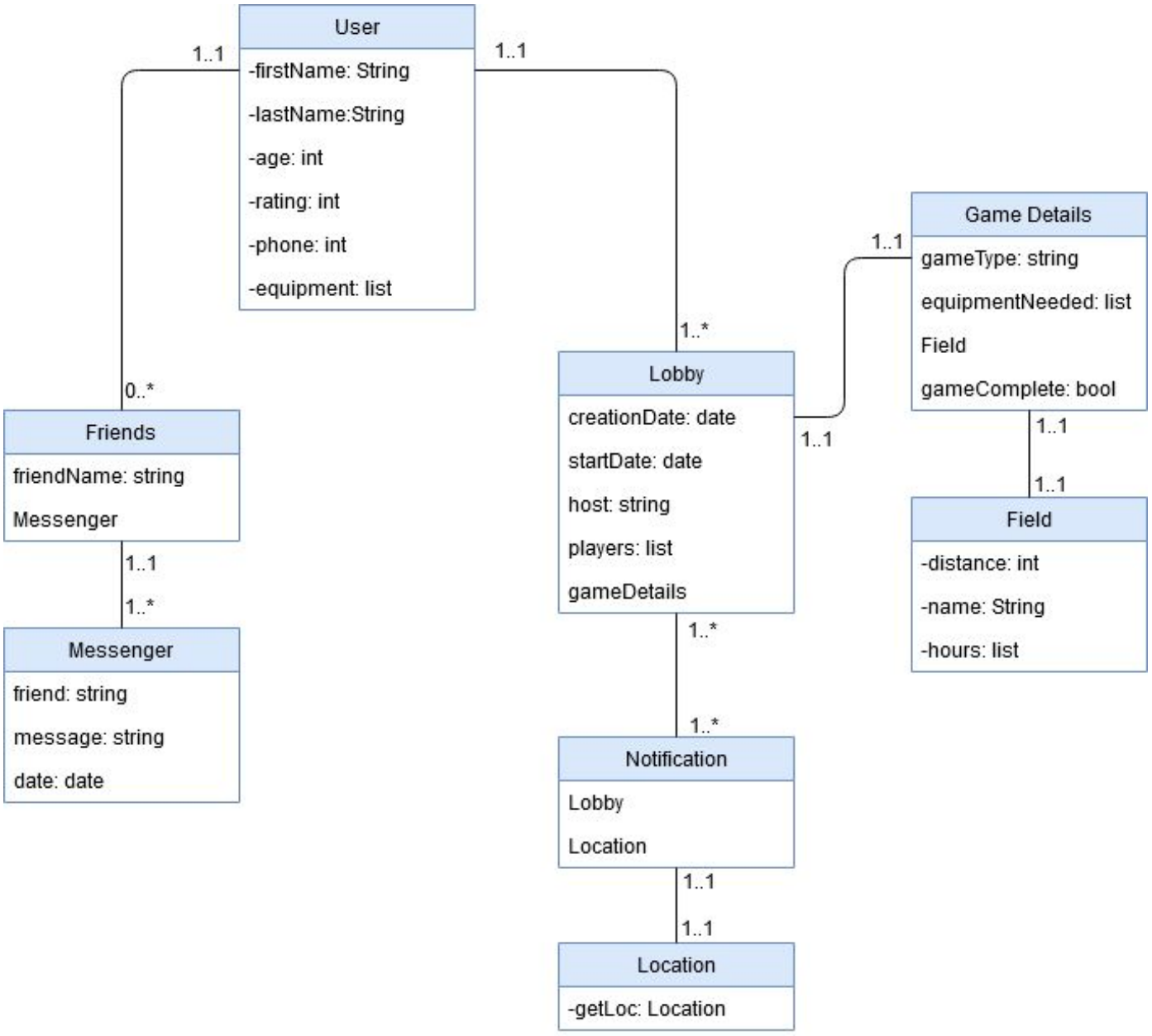
Activity Diagram



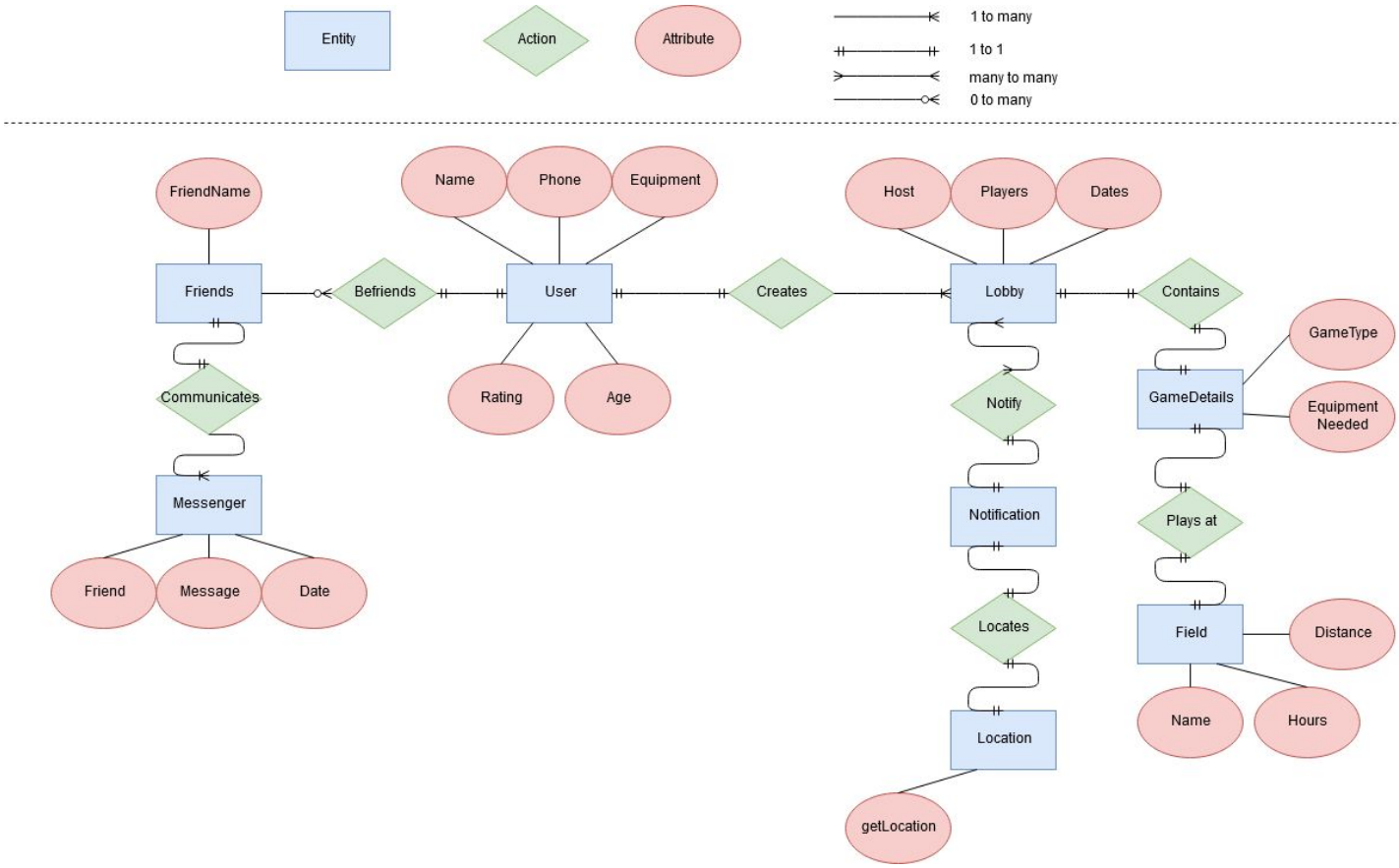
Sequence Diagram



Class Diagram



Relational Diagram



Analysis

Tradeoff Analysis

Criteria	Metrics	Weight	Max Value
Major Architecure Pattern(s)	Interpreter Pattern (SQL)	10%	25
	Client-Server Pattern (message, email)		25
	Layered Pattern (General Desktop Apps)		50
Total			100
Component Choices	UI	20%	10
	Client		15
	Server		15
	Database		20
	Machine Learning		40
Total			100
Language Choices	JavaScript	15%	10
	Java		20
	Python		50
	PHP		20
Total			100
Framework Choices	Data-Driven Framework	15%	25
	Hybrid Driven Testing Framework		50
	Behavior Driven Testing Framework		25
Total			100
Database Choices	MySQL	5%	100
Total			100
Server or Serverless Choices	MySQL Server	5%	100
Total			100
Front End Framework Choices	React	5%	100
Total			100
API Choices	Google Sign In	25%	50
	Google Maps		50
Total			100
Final Total		100%	

Machine Learning Write Up

Application Use Case
Whenever someone wants to join or host a pick up game ahead of time without the hassle of going to the nearest park where there are uncertain amount of people there willing to play a game with them.

Stakeholder's Benefits
Stakeholders will not waste time nor will they need to go far to find a pick up game. In addition, with the use of machine learning, users will find games that match their likings. This will result in better experience and faster networking.

ML 10 Steps Documentation
1. Gather answers from questions provided from signing up.
2. Gather statistic data from previous host/join games
3. Gather the amount of "joined" events
4. Gather the amount of "hosted" events
5. Gather experience
6. Gather location of events
7. Display games that were compiled from the data set
8. Gather how many games from the displayed games were actually attended and had positive experience
9. Learn from the gathered games to output better events
10. Reiterate through these 10 steps.

ML Model vs. Our Objectives
Our machine learning model that we plan to use is supervised learning. It consists of a target/outcome variable which is to be predicted from a given set of predictors. Using these set of variables, we generate a function that maps inputs to desired outputs. The training process continues until the model achieves a desired level of accuracy on the training data. Since our objective for the application of machine learning in our product is to find pickup games that are best suited for our users, this model works hand-to-hand with our objectives

Architectural Choices
- Web Server
    - We chose to use an interpreter pattern and client-server pattern architecture since it works well with our database, which is based on SQL, and messaging system, such as user-to-user communication and system messages. We also chose to use a layered pattern architecture since it works well with mobile applications.