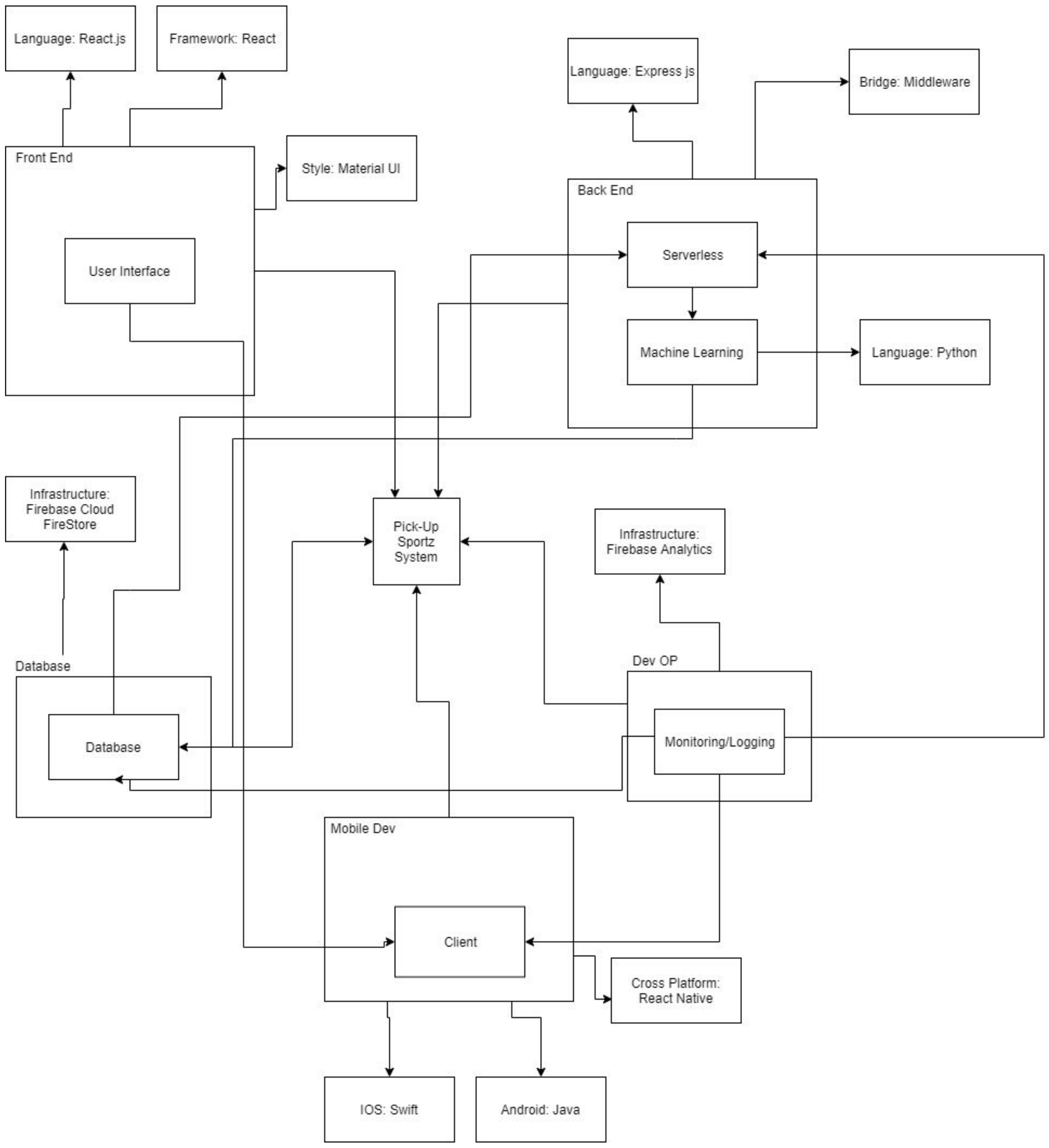
Pick-Up Sportz
Architecture and Design

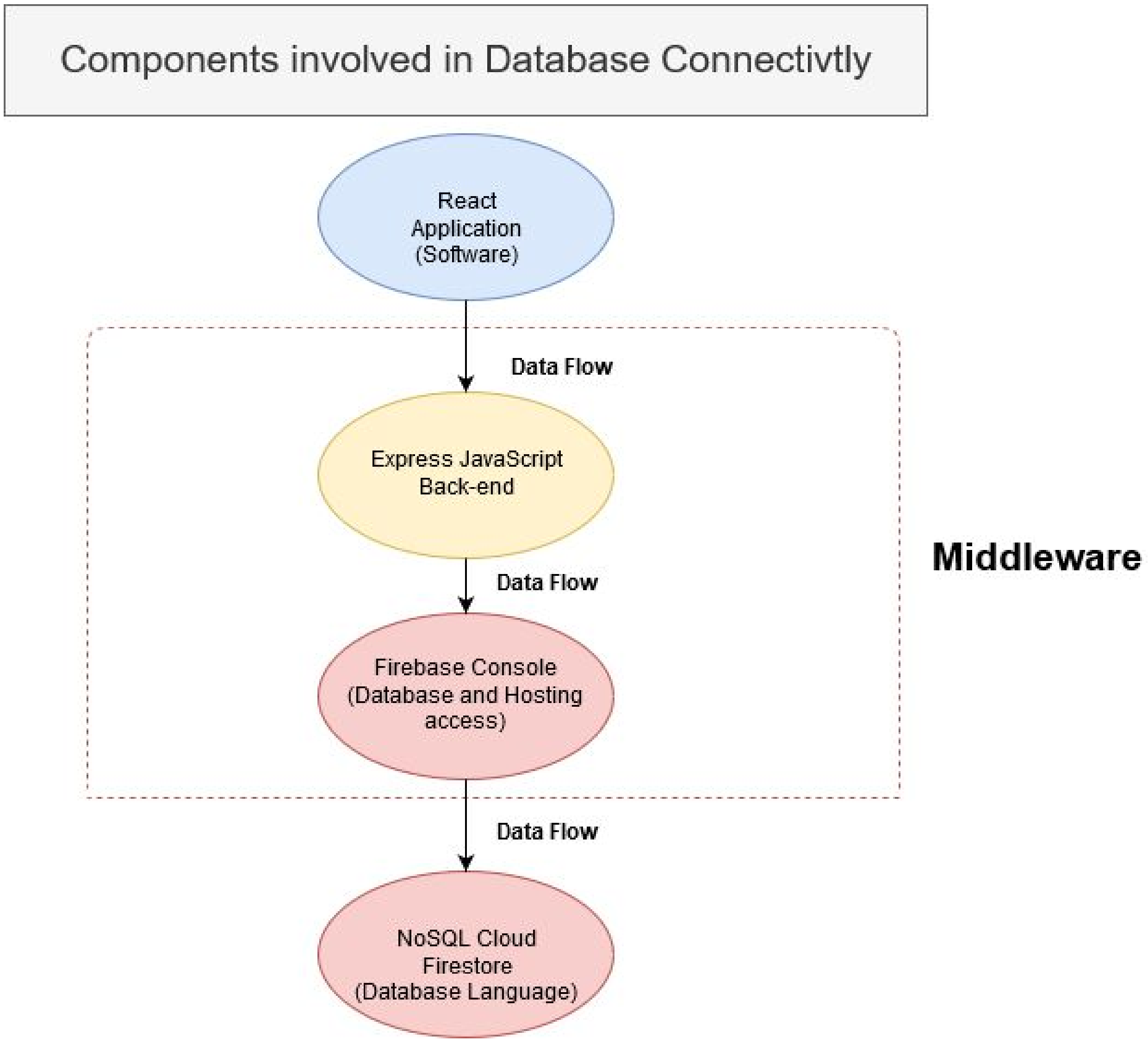
By
John Him
Jamil Khan
Brandon Le
Benjamin Seo
Chaz Del Prato
Christine Duong

Table of Content

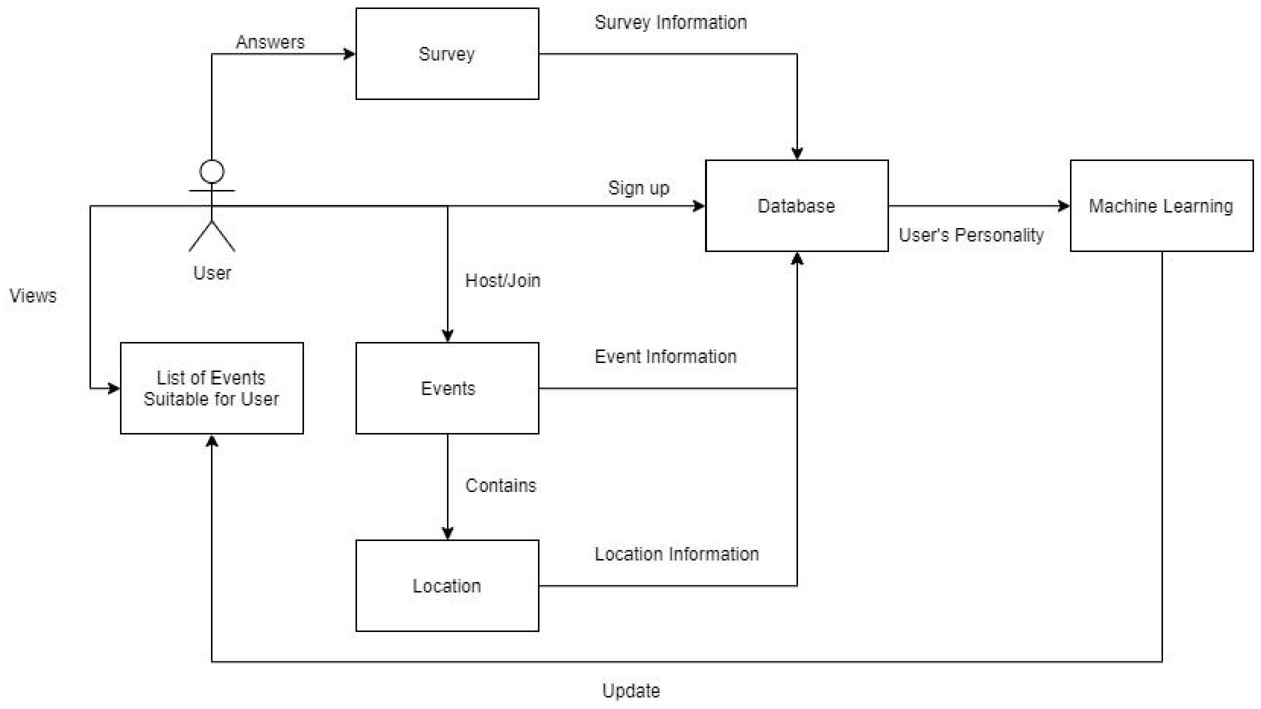
Diagrams

System Diagram(s) and Component Diagram(s)




Components involved in Database Connectivtly
React Application (Software)
Data Flow
Express JavaScript Back-end
Data Flow
Firebase Console (Database and Hosting access)
Middleware
Data Flow
NoSQL Cloud Firestore (Database Language)

Information Flow Diagram



Quality and Quantity Standard

To set up the front end of our web application, we will be using React.js 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.

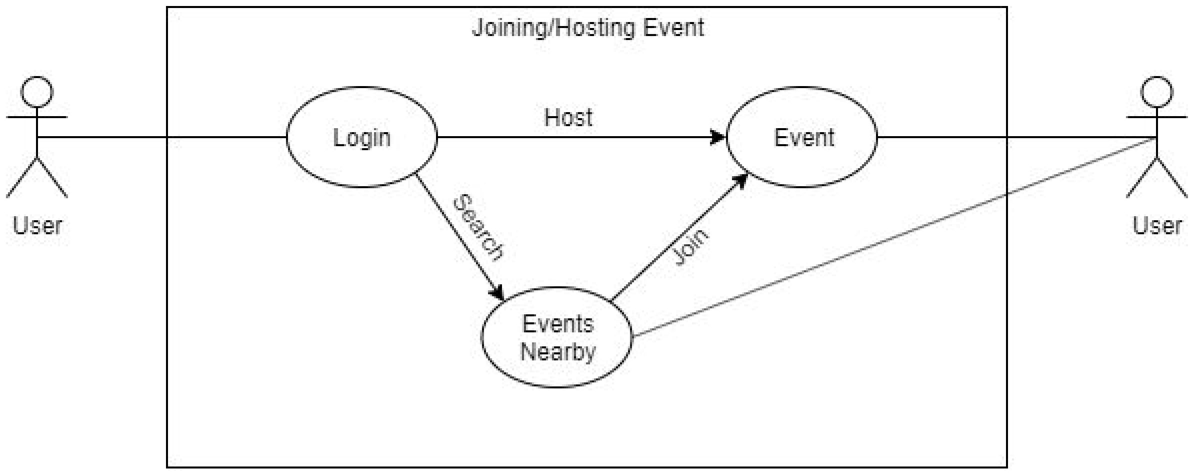
To set up the back end of our application, the language we are planning to use is Express.js. This will consist of a serverless component and our machine learning component written in Python that will communicate with our client, database, and our DevOps.To bridge our serverless component to our database we would use a middleware framework.

To set up the database component, we will use NoSQL Cloud Firestore 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 Firebase with a firebase analytics. This portion will consist of our monitoring/logging components that will communicate with our client, server, and database.

Use-Case Diagrams

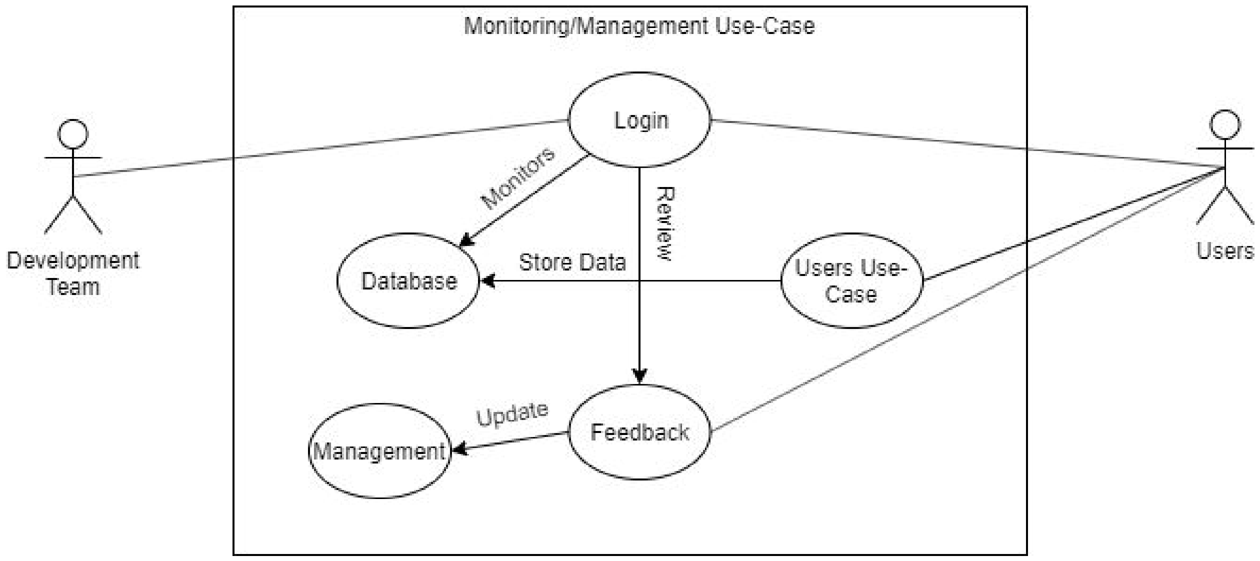
User Host/Join Use Case:



- 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:

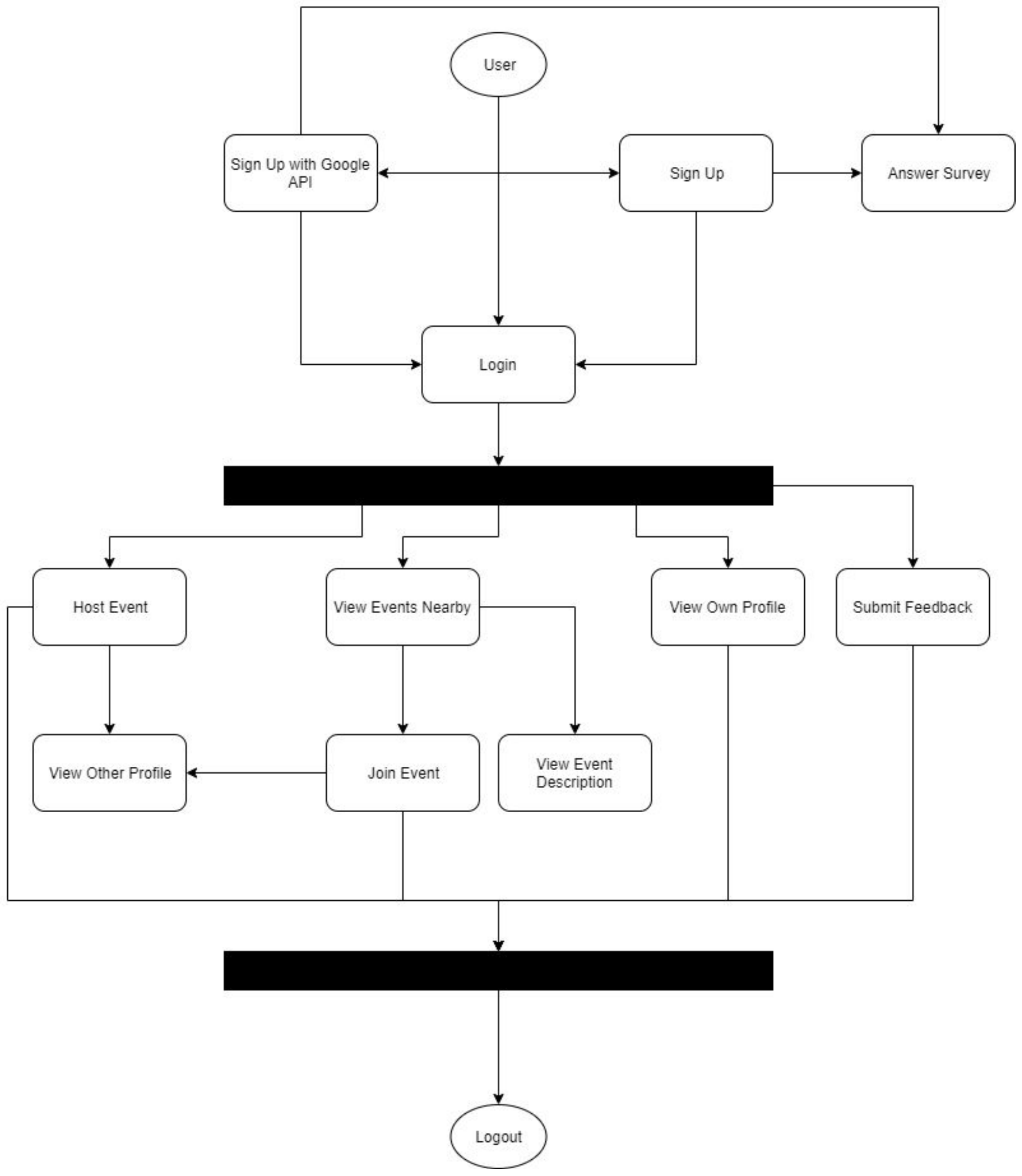


- Use Case Field
    - This use case shows what and how the development team monitors/manage in the system
- Use Case Name
    - Manage/Monitor the System
- Subject Area
    - Developers Use-Case
- Business Event
    - Login
- Actors
    - Development Team
    - Users
- Use-Case overview
    - The development 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

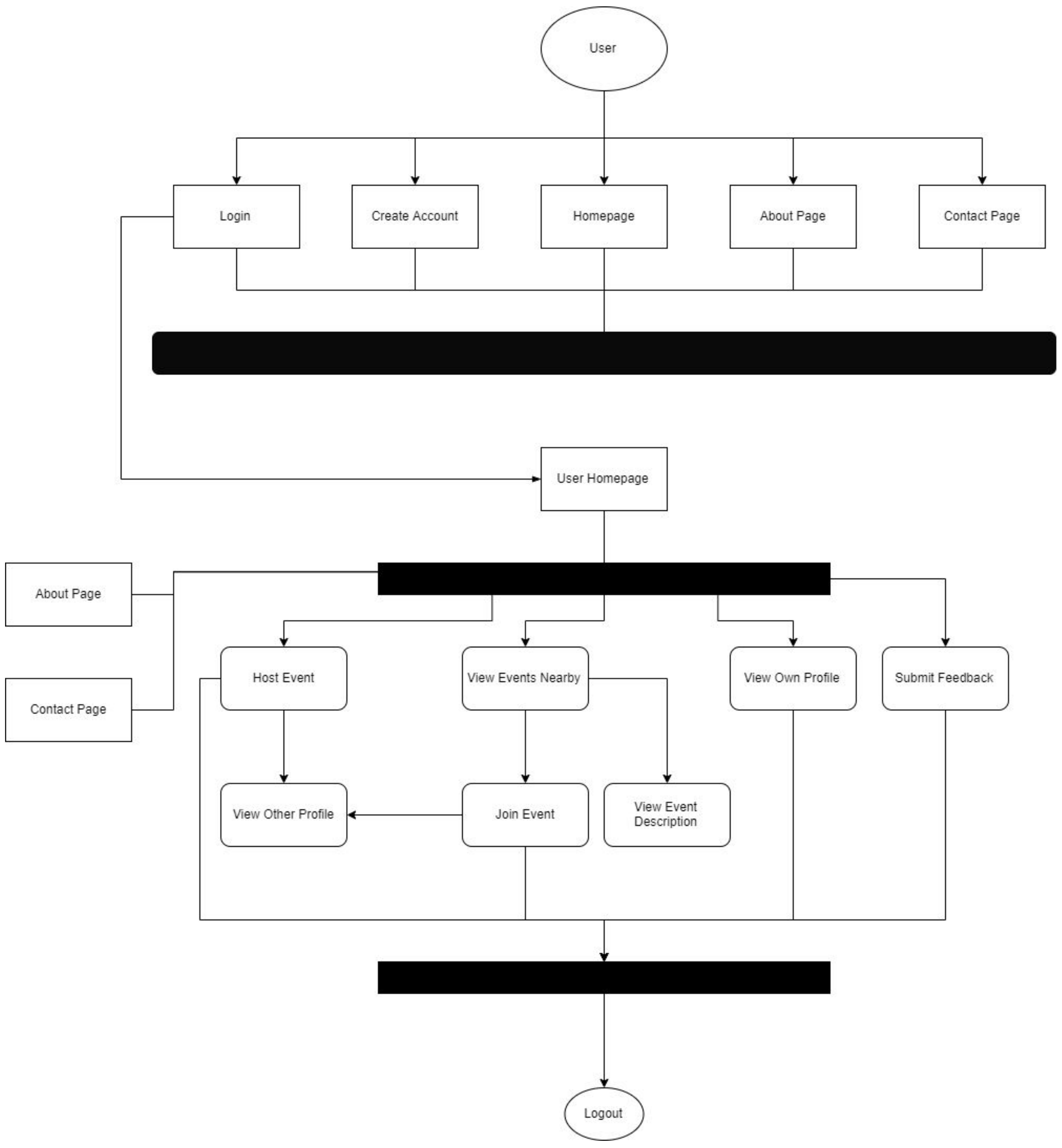
    - Login information
    - Notification for change in system
    - Update Notification
    - Feedback reply
- Output summary
    - System Notification
    - Feedback reply
- Usability Index (out of 10)
    - Satisfaction (6)
    - Importance (10)
    - Frequency (10)
- Use case notes
    - This use case 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

Mobile

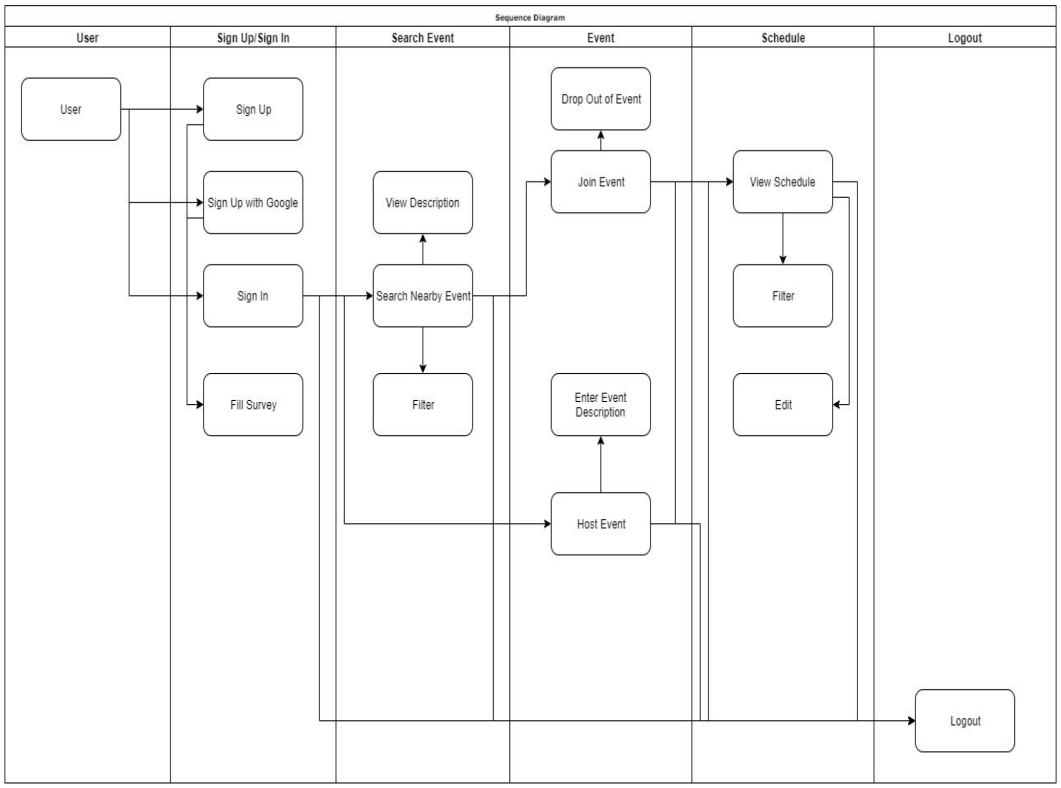


Web

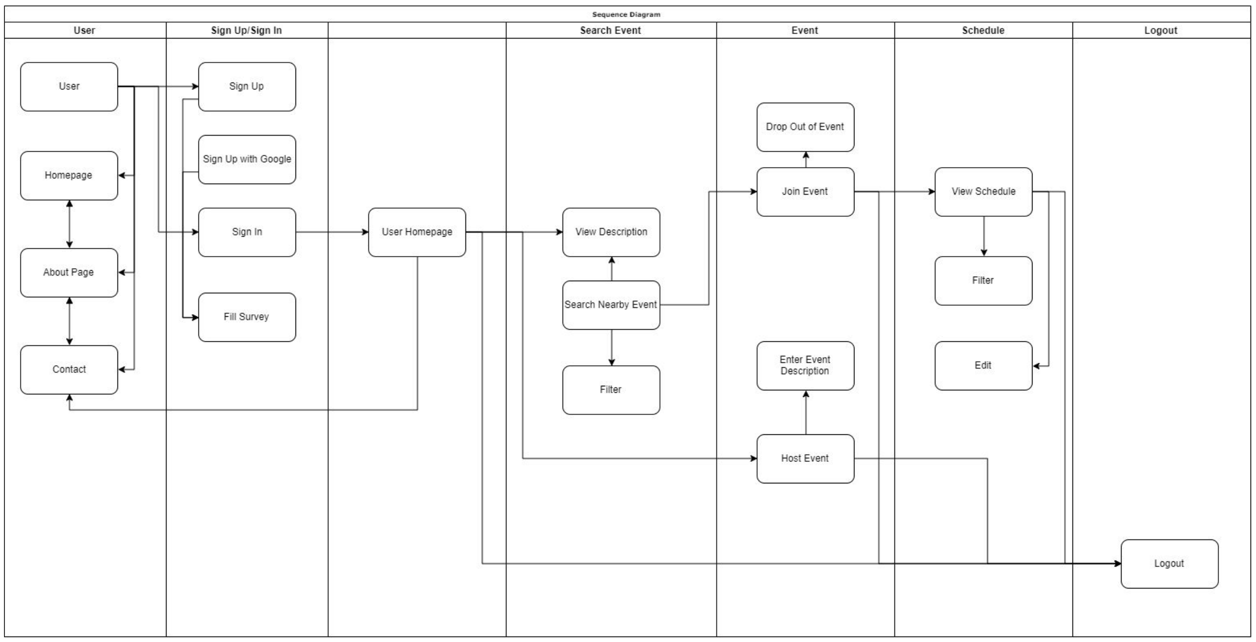

User
Login
Create Account
Homepage
About Page
Contact Page
User Homepage
About Page
Contact Page
Host Event
View Events Nearby
View Own Profile
Submit Feedback
View Other Profile
Join Event
View Event Description
Logout

Sequence Diagram

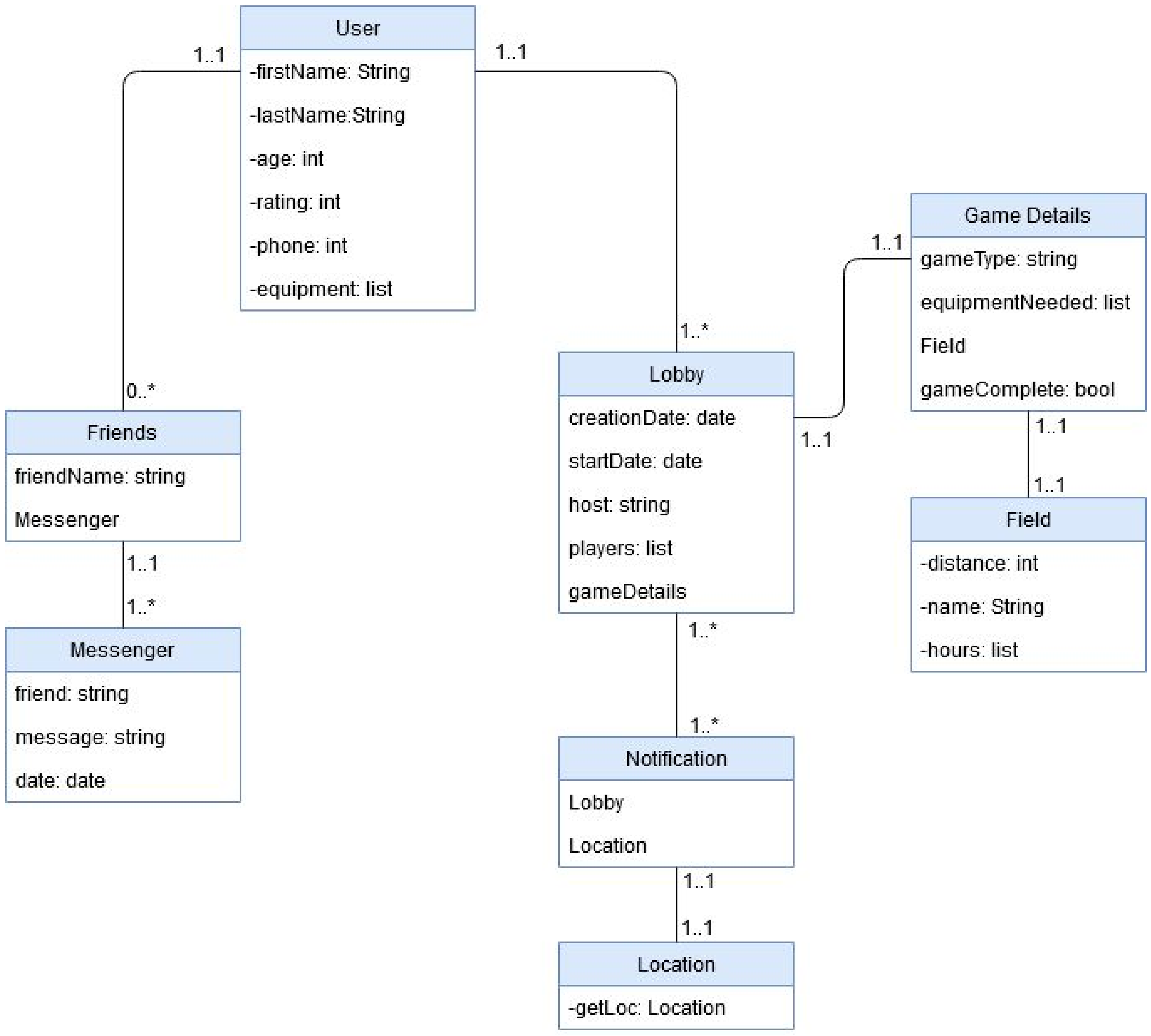
Mobile



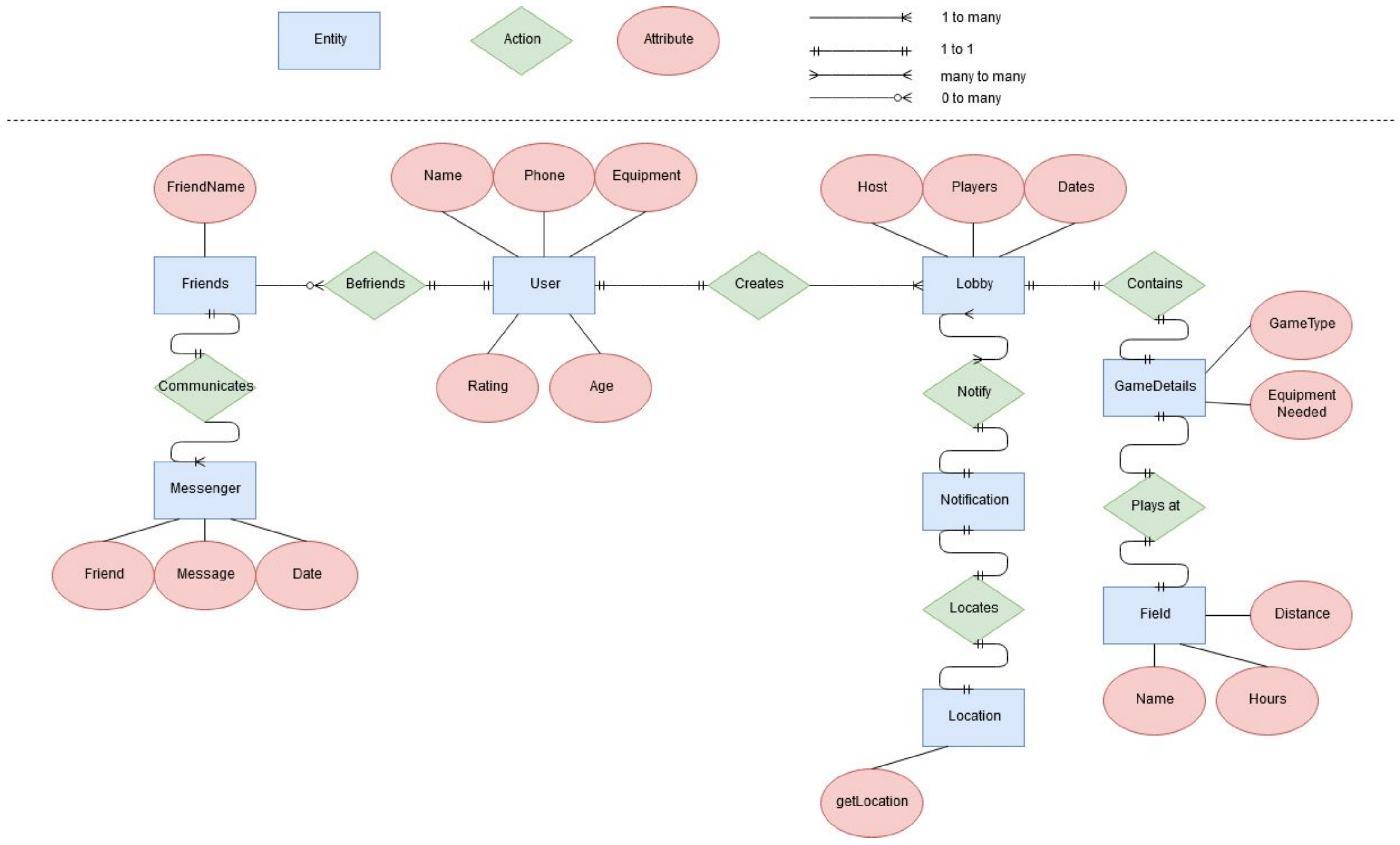
Web



Class Diagram



Relational Diagram



Analysis

Tradeoff Analysis

Architecture Patterns Employed	Pros	Cons
Layered Pattern	Lower layer can be used by different higher layers. Layers make standardization easier as we can clearly define levels. Changes can be made within the layer without affecting other layers	Not universally applicable. Certain layers may have to be skipped in certain situations.
Client-Server Pattern	Good to model a set of services where clients can request them.	Requests are typically handled in separate threads on the server. Inter-process communication causes overhead as different clients have different representations.
Master-Slave Pattern	Accuracy - The execution of a service is delegated to different slaves, with different implementations.	The slaves are isolated; there is no shared state. The latency in the master-slave communication can be an issue, for instance real-time systems. This pattern can only be applied to a problem that can be decomposed.

Component Choices	Pros	Cons
Front End		
Languages		
HTML	Publish Online Docs, Retrieve online info, designing forms, include apps in their documents. Easy to learn.	Open to author interpretation. Technical progress is slow. It is a declarative language, meaning that it has limited functional prowess

	Well-interpreted by browsers. HTML parsers are forgiving. Light-weight. Has a vast array of structural and aesthetic elements that can infer meaning and presentation.	compared to functional languages.
CSS	Design colors, fonts and layouts. Adapt display across platforms. Easier site maintenance. Tailored pages. Help make spontaneous and consistent changes. Improves page loading speed. Device Compatibility. Ability to reposition. Makes the search engine better crawl your web pages.	Cross-browser issues. Confusion due to its many layers. Vulnerable
Javascript	Speed, simplicity, popularity, interoperability, server load, and gives the ability to create rich interfaces.	Client-side security. Different browsers may use javascript.
Framework		
AngularJS	Two-way data binding, DOM manipulation, improved server performance, faster application prototyping, responsive web, MVVM architecture, plain html templates, fast development	JavaScript support is mandatory, inexperience with MVC, difficult to debug the scopes. Difficult to learn.
ReactJS	Updates process is optimised and accelerated. JSX makes components/blocks code readable. It displays how components are plugged or combined with. React's data binding establishes conditions for creation dynamic applications. Up to date. Facebook team supports the library. Advice or code samples can be given by Facebook community.	Learning curve. Being not full-featured framework it is required in-depth knowledge for integration user interface free library into MVC framework. View-orientedness is one of the cons of ReactJS. It should be found 'Model' and 'Controller' to resolve 'View' problem. Not using isomorphic approach to exploit application leads to search engines indexing problems.
JQuery	jQuery is flexible and fast for	jQuery is easy to install and learn,

	web development. It comes with an MIT license and is Open Source. It has an excellent support community. It has Plugins. Bugs are resolved quickly. Excellent integration with AJAX.	initially. But it’s not that easy if we compare it with CSS. If jQuery is improperly implemented as a Framework, the development environment can get out of control.
Style		
Bootstrap	You are the Boss. You Pick the Focus. You Maintain Responsibility.	Personal Risk. Lack of Networking. Slow Growth.
Material UI	Included icon font. Customizable. Responsive. Large selection of components. Perfect for React projects. Adheres really well to the material design standard.	Large / heavy. Active development means API changes are frequent. Refuses to use the flexbox model.
Back End		
Languages		
Express.js	flexibility, simplicity, extensibility and performance. Fast app development. I/O request handling. Open-source community. Easy integration of third-party services and middleware. Easy to learn.	Event-driven nature (callbacks). Code organization.
Python	Very good with dealing with AI and ML. Preferred languages for creating AI or ML based web & mobile apps. A multipurpose language which can be used by software product development. Easy to use.	Performance issue. Absence from mobile and browser computing. A lot of design restrictions.
Ruby	Oo language , Excellent documentation, Rapid prototyping/ Development, RoR=MVC framework, Easy language to start with , Cool community (extremely	Not that hot anymore , Ruby generally except for web apps is not too notch (mainly comparing with python) Generally regarded as a slow language.

	friendly)	
Database		
Infrastructure		
MySql	.MySQL products remain solid. There is more MySQL investment and innovation than ever before. MySQL is designed with a focus on the Web, Cloud and Big Data. There are more MySQL projects than before.	MySQL is an open source (kind of). MySQL is not as mature as other relational database management systems. MySQL is Oracle-owned instead of community driven. Big names are jumping ship
Firebase	simple serialization of app state. 3-way data binding via Angularfire. minimal setup. no server infrastructure needed to power apps with data.	not widely used or battle tested for Enterprises. very limited querying and indexing. no aggregation. no map reduce. can’t query or list users or stored files.
DevOP		
Infrastructure		
AWS	Easy to Use. No Capacity Limits. Provides Speed and Agility. Secure and Reliable.	Limitations OF Amazon EC2. Security Limitations. Technical Support Fee. AWS pricing list. General cloud Computing Issues.
React	Biggest community and code owned and maintained by Facebook. Big ecosystem. Flexibility. Scalable. Can be used for native and web apps.	Requires to learn JSX and is more programmer-oriented. It’s kind of verbose. Writing components isn’t as straightforward as pure HTML & JS. Being too flexible in structure can be problematic. In React, everything is just JavaScript.
Vue	Backed by Laravel and Alibaba. Big players are starting to adopt the framework. Simplicity of the syntax and short learning curve for newcomers. Uncomplicated structure. Features a lot of concepts from Angular 1 and React.	Owned by one person, maintained by a small team. Being too flexible in structure can be problematic. Legacy code from the current Angular 1.5 will need some conversion.

Mobile Dev		
Android		
Java	Used for building enterprise scale web application. Android mobile app developers also rely on this language. Most stable language.	Paid for commercial use. Poor performance. Far from a native look and feel on the desktop. Verbose and complex code
SDK	Easier integration. Faster time to market. Stronger security. Cost savings. Reliability.	Unfamiliar
IOS		
Objective C	Objective-C is a mature language with a huge community, lots of experienced developers, best practices, and coding styles. There are plenty of third-party libs, that are well tested, used in many huge projects. (note: some of them may not be updated to the latest iOS releases). There are tons of legacy code that needs to be supported. (This one especially for Crusty). Compatibility with other languages. You can use C or C++ with no problem inside your Objective-C files. If you need to use C++ in your Swift code you should first wrap it in Objective-C code first and then import that in Swift.	Sooner or later Objective-C won’t be able to support some new features, so there will be Swift-only marked methods and frameworks. It is an outdated language with lack of many valuable features available in other programming languages.(Just look at Swift enums and cry).
Swift	Modern, more expressive, feature-rich programming language. Very active community. Ported to other platforms. Rapidly applying new features and proposals.	lack of 3-d party solutions due to fast evolution of language. There are lots of projects on Github that still on Swift 1.2. Immature community and a lot of misleading advice.
Cross Platform		
React Native	Faster to Build. One	Less Smooth Navigation. Lack of

	Framework, Multiple Platforms. Hot Reloading. Smaller Teams. Fast Applications. Simplified UI.	Some Custom Modules. Native Developers Still Needed.
PWA	Progressive. Responsive. App-like. Updated. Secure. Searchable. Reactivable. Installable.	iOS support from version 11.3 onwards. greater use of the device battery. not all devices support the full range of PWA features. it is not possible to establish a strong re-engagement for iOS users. support for offline execution is however limited.

Language Choices	Pros	Cons
Java	Used for building enterprise scale web application. Android mobile app developers also rely on this language. Most stable language.	Paid for commercial use. Poor performance. Far from a native look and feel on the desktop. Verbose and complex code
Python	Very good with dealing with AI and ML. Preferred languages for creating AI or ML based web & mobile apps. A multipurpose language which can be used by software product development. Easy to use.	Performance issue. Absence from mobile and browser computing. A lot of design restrictions.
Java Script	Fast, simple, popular, interoperability, serverload, rich interfaces, extended functionalities, versatile.	Client-side security. Different browser support.
React.js	Components can be reused. It has a virtual DOM. Popular.	Fast pace in development. Lack of conventions. Difficult to learn
Express.js	Flexible, simple, extensible, and good performance. Fast app development. Good I/O request handling. Open source community. Easy integration of 3rd party	Performance bottlenecks with heavy computation tasks. A lot of callback issues.

	services and middleware	

Framework Choices	Pros	Cons
AngularJS	A complete framework. 2D Data flow. A complete solution. Improved server performance. Faster application prototyping. Responsive Web. Plain HTML templates	Difficult to learn overall. JavaScript is mandatory. Inexperience with MVC. Possible time consumption
ReactJS	Easy to learn and use. Creating Dynamic Web applications is easier. Reusable components. Performance enhancement.	Only a javascript library. Difficult to learn if working with Redux. 1D data flow. High pace of development. Poor Documentation. JSX as a barrier
Django	Easy to learn. Clarity and Readable. Versatile. Fast to write. No Design Holes. Fast. Fully loaded. Secure. Scalable.	Uses routing pattern to specify its URL. Too monolithic. Everything is based on Django ORM. Components get deployed together. Need to know the whole system to make it work.

Database Choices	Pros	Cons
SQL/RDBMS/Relational	Relational Databases are well-documented and mature technologies, and RDBMSs are sold and maintained by a number of established corporations. SQL standards are well-defined and commonly accepted. A large pool of qualified developers have experience with SQL and RDBMS. All RDBMS are ACID-compliant - they satisfy the requirements of Atomicity, Consistency, Isolation, and Durability.	RDBMS do not work well, or at all, with unstructured or semi-structured data due to schema and type constraints. This makes them ill-suited for large analytics or IoT event loads. The tables in the relational database will not necessarily map one-to-one with an object or class representing the same data. When migrating one RDBMS to another, schemas and types must generally be identical between source and

		destination tables for migration to work (schema constraints). For many of the same reasons, extremely complex datasets or those containing variable-length records are generally difficult to handle with RDBMS schema.
NoSQL/Non-Relational Databases	Schema-free data models are more flexible and easier to administer. NoSQL databases are generally more horizontally scalable and fault-tolerant. Data can easily be distributed across different nodes. To improve availability and/or partition tolerance, you can choose that data on some nodes to be eventually consistent.	NoSQL databases are generally less widely adopted and mature than RDBMS solutions, so specific expertise is often required. There is a range of formats and constraints specific to each database type.

Server vs. Serverless Choices	Pros	Cons
Server	Have access to a lot of public APIs. Can handle complex and long-running codes faster and easier. It is scalable.	Cost more. Difficult to set up different environments.
Serverless	Cost Less. Easier to set up different environments. It is scalable	Can only access private APIs. Complex and long-running functions are not good.

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 (Layered Pattern Architecture)
    - We chose to use a layered pattern architecture since it works well with our database, which is firebased, and messaging system, such as user-to-user communication and system messages. It also works well with web applications.