Architecture and Design Document: Quikpik

California State University, Long Beach

CECS 491A-Sec 11 Spring 2020

March 16th, 2020

Ernie Argel 017984237
Jalon Flores 015843540
Ranjit John 016695989
Anthony Pham 014415919
Judy Tran 017194591
Juan Villa 015909255

Table of Contents

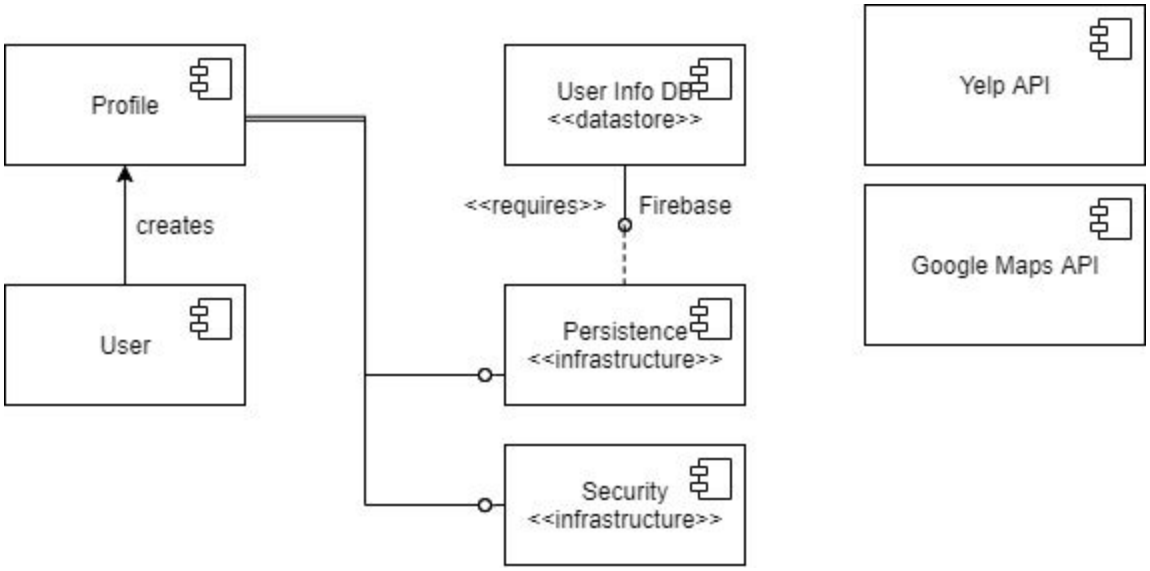
Version History

Version No.	Date
1.0	March 16, 2020

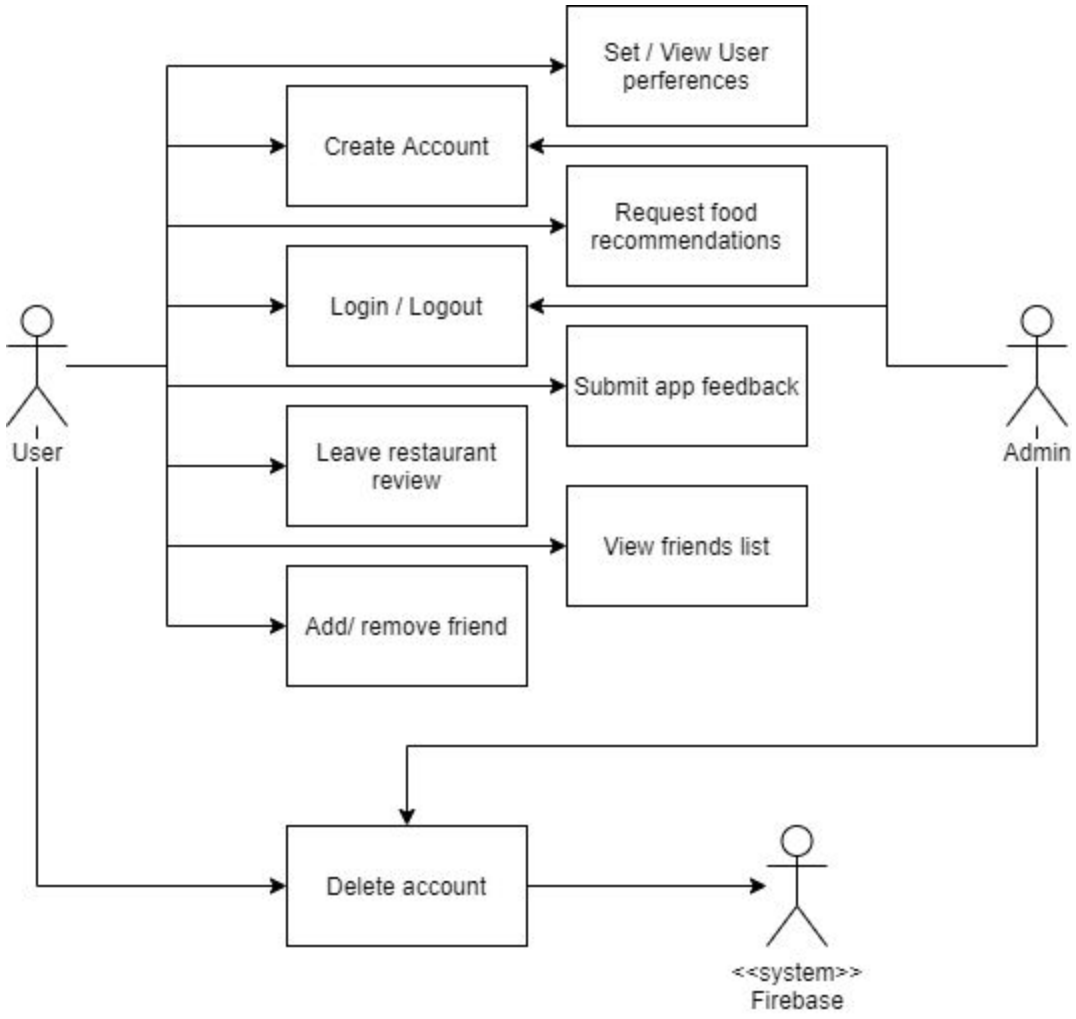
2.0	April 6, 2020

Diagrams

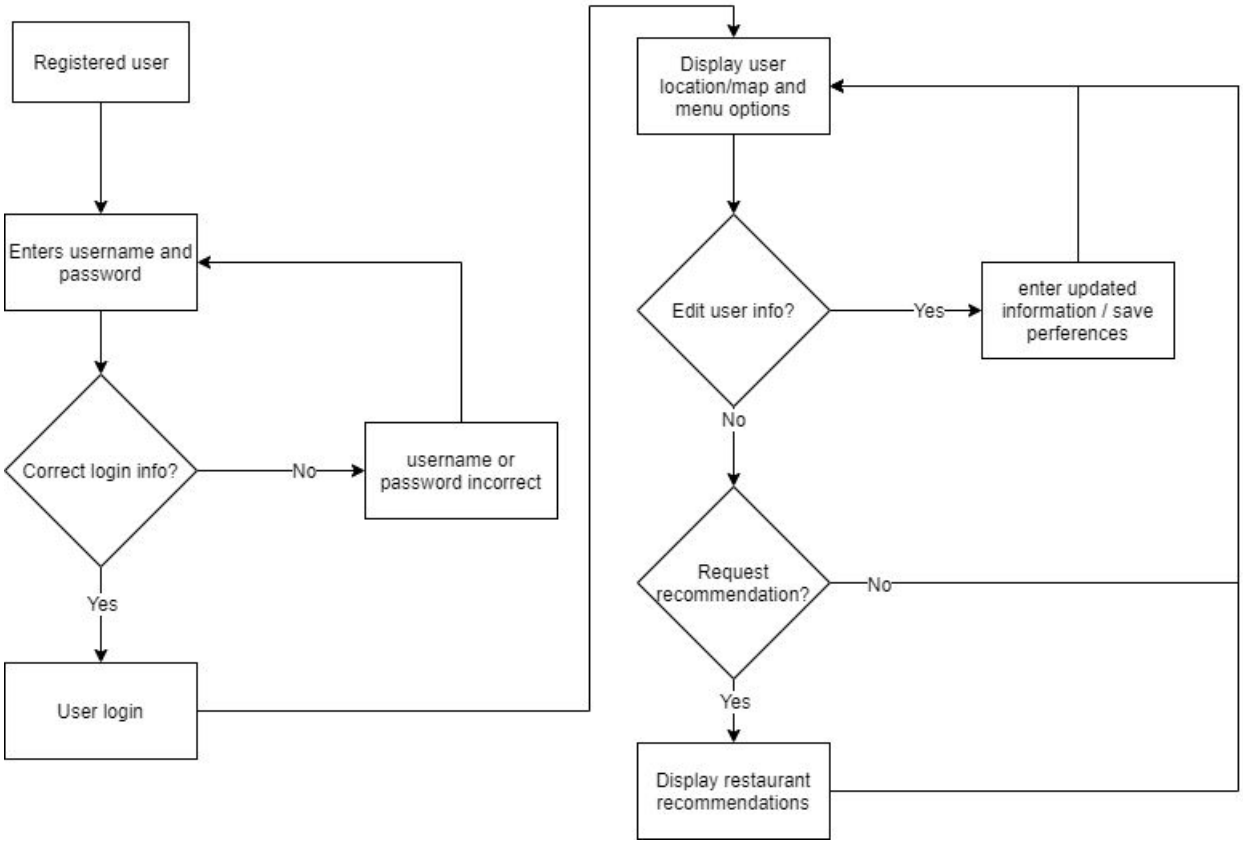
System Component Diagram



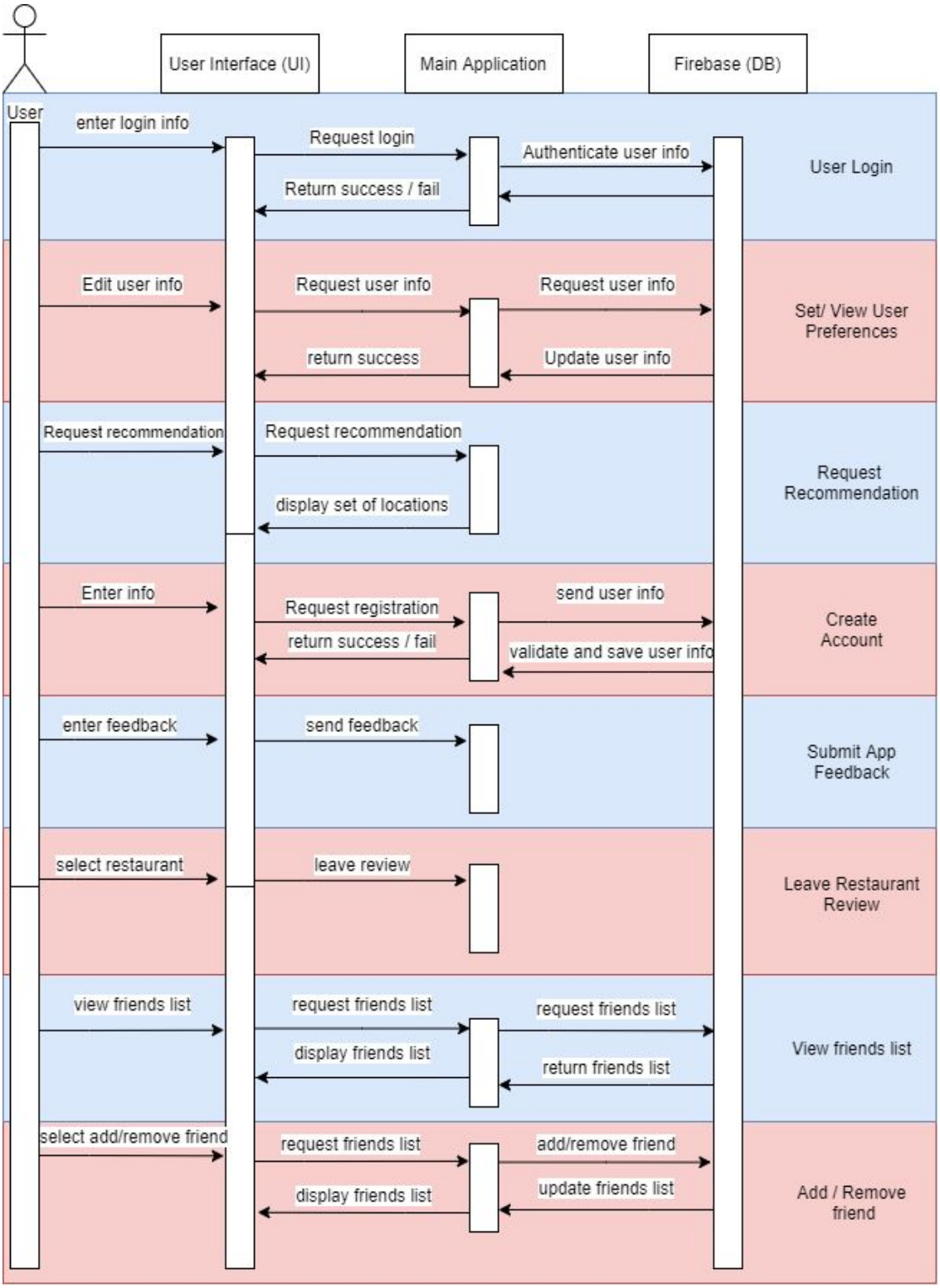
Use Case Diagram



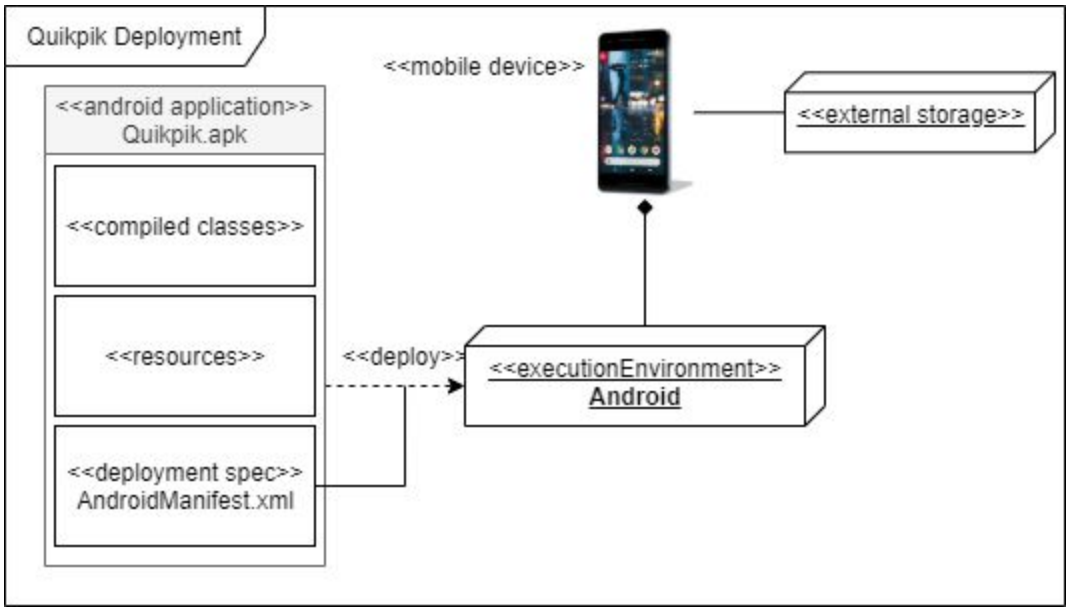
Activity Diagram



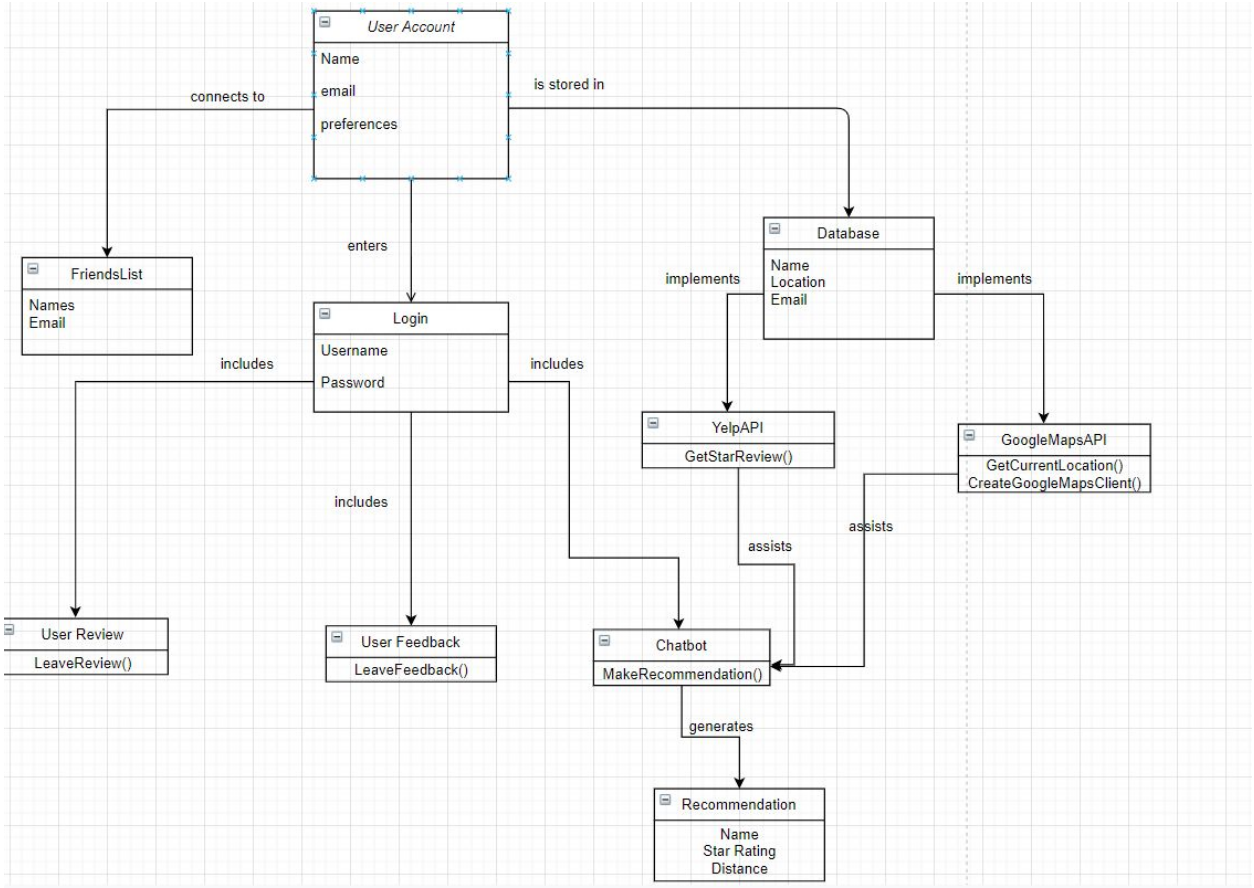
Sequence Diagram



Deployment Diagram



Class Diagram



Tradeoff Analysis

Frameworks			
Criteria	**Android Studio**	**React**	**Angular**
Team Familiarity	2	0	0
Time Constraint	1	0	-1
Scalability	-1	2	1
Maintainability	1	2	1
Performance	-1	1	2
Portability	0	1	1
Testing	0	-1	1
Flexibility	1	1	0

Learning Curve	1	-1	-2
Sum +'s	6	8	6
Sum 0's	2	2	2
Sum -'s	2	2	3
Net Score	4	6	3
Rank	2	1	3

Conclusion:

Even though "React" might be the best for implementing our application base on scalability, maintainability, and flexibility, we believe "Android Studio" would be the better choice in terms of team familiarity and the time constraint of developing our first iteration of our app. Since we are trying to develop a mobile application, performance will be smoother and responsive for our users. Also, we are able to use the built-in features in the user's device like retrieving their location to find the best restaurant near them.

Database			
Criteria	**Firebase**	**MongoDB**	**PostgreSQL**
Team Familiarity	1	0	0
Time Constraint	1	-1	-1
Maintainability	0	1	1
Flexibility	1	1	-1
Scalability	1	2	1
Authentication	1	1	0
Security	1	2	0
Performance	1	2	1
Cost	-1	-1	1

Data Synchronization	1	1	1
APIs	1	1	0
Supported Languages	1	1	1
Learning Curve	1	-1	1
Sum +’s	11	14	7
Sum 0’s	2	1	4
Sum -’s	1	3	2
Net Score	10	11	5
Rank	2	1	3

Architecture Pattern (Client-Server Model)

Describes the relationship of each program communicating within the application. The server component is the one that provides the functions or service to the client(s) and initiates requests for the services.

Client

- As for our client-side we are using Android Studio as means for users interacting with our application. The client is provided with a UI they can interact with and for each input there is an output.

Server

- As for our server-side we are using Firebase to provide real-time connectivity and data synchronization.

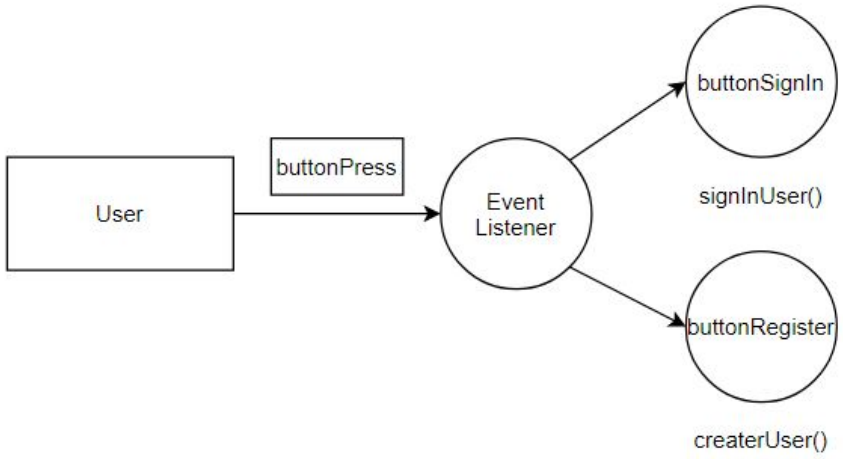
For each screen interaction by the user and if a data is changed, a request is initiated to our server-side which is Firebase, then the data is synchronized across all users and a response is sent back when data is changed. For example, when a client/user initiates a request (send a friend request) to our server, our server responds by synchronizing data to the other requested client and if the other client accepts the request, the server would add both clients to their friends list. Even though our app is serverless it still acts as a cloud platform that provides real-time communication between other clients and shares resources across all clients.

Command Pattern (Behavioral Design Pattern)

Encapsulates the request as an object which let's us parameterize different requests (log requests). Also it supports undoable operations which means when initiates a command they cannot undo the action they have set except with another command.

For our application we implemented a button interface that has different parameters and operations when initiated.

Ex: buttonSignIn and buttonRegister implement the button method setOnClickListener() where it executes the different operations within that method. The buttonSignIn signs in the given parameters (email, password) while the buttonRegister authenticates the user and creates the user in our server.



Framework Choice - Web vs. Mobile Framework

Web

Pros:

- Portability
    - Easier to distribute to users; do not need to be downloaded in the app store.
- Does not take up storage on the user's device.
- Don't have to worry about if the app needs to be updated.
- Can be built easily with the available templates online

Cons:

- Not much support across all mobile browsers.
- Usability issues: load times, network availability, etc.
- Slower and less responsive.

Mobile:

Pros:

- Better performance on user devices.
- Smooth and interactive in terms of user input and output.
- Can work with built-in features of the user's device.

Cons:

- Require experienced developers.
- Costly.
- Mobile platforms each have unique development processes.

Conclusion:

Developing a web app may be easier when it comes to developing a smaller application and deploying it to users. However, may have usability issues when it comes to user interactions. In conclusion we chose to develop a mobile app because our goal is to get our app on user devices and work with the built-in features in the devices like their location. This is the reason why we chose to adopt Android Studio as our first framework to develop our app.

Server vs. Serverless

Server:

Pros:

- Full access
    - Have control over the server 24/7
- Performance boost and can save on energy costs.
- More secure and are taken with greater precautions as no one else has access to your server

Cons:

- Cost
    - Hosting a server may cost more to maintain than the cloud.

Serverless:

Pros:

- Cheaper and easier to use and offer more services.
- Cost may vary, but only pay for the services and storage.
- Can share resources from any location to any device.

Conclusion:

Using a server may be better for more control and performance. However, since we are poor college students, the cost to maintain might be out of our budget. In the end, we chose the serverless approach because it is easier and cheaper to implement. Also, this platform is growing more and more, and is being recognised by organizations and gaining support as a result. This is the reason why we chose to use “Firebase” as our server, as it is serverless and we only have to pay for services and additional storage.

Risk Management Table

About:

Our group has identified some possible risks that might affect us in the development of our app. The table below is divided into six sections: id(identity number of the risk), description(What the risk is), mitigation scheme(plan to combat/avoid the risk), security level, date identified and status.

Severity Levels:

High - high impact on the app if risk occurs.

Medium - moderate impact on the app if risk occurs.

Low - little or no impact on the app if risk occurs.

Status Level:

Ongoing - Members are currently addressing the risk.

Complete - Members addressed the risk and took necessary steps to mitigate the risk.

Incomplete - Members have not addressed the risk.

ID	Description	Mitigation Scheme	Severity Level	Date of Identification	Status
R1	User information database falls victim to SQL injection attack and we lose user data.	Using parameterized statements whenever possible when creating login and password in database	**High**	02/27/20	Ongoing
R2	User password not encrypted and visible in database	Using hash functions or some sort of encryption to lock away user passwords	**High**	02/27/20	Ongoing
R3	Inadequate machine learning model for chatbot	Constant research on classical machine learning models	**High**	03/01/20	Ongoing
R4	Chatbot doesn't recognize speech	Think about a Semantic analysis based model. We always revert to secondary design of text based bot.	**Med**	03/01/20	Ongoing
R5	Chatbot isn't conversational	Constantly researching a machine learning model that uses user chat, profile, and yelp review data as a corpus for ML model. We can always revert to secondary design of text based bot.	**Med-High**	03/01/20	Ongoing
R6	User suggestion isn't personalized	Implement the Yelp API so we have a backup suggestion style if ours fail	**Med**	03/01/20	Ongoing
R7	Activity monitoring and data retrieval issues	Creating a centralized database that is efficient in storing and retrieving user information.	**High**	03/03/20	Ongoing
R8	Version of user device is	This app can be run on 99.8% of Android devices.	**Low**	03/03/20	Incomplete

	outdated to run the app	Android Studio has this option to control which devices can run the project.			
R9	Users won't like some aspects of the app	Implement a feedback hub that the customer can write their concerns on.	**Med**	03/04/20	Incomplete
R10	Slowed development from group because of other classes	Assign coding goals and assignments in pairs. So if one member is busy the other can still work	**Med**	03/07/20	Complete
R11	Performance on the app is too slow	Optimize device performance by monitoring the back-end effectively	**Low-Med**	03/07/20	Incomplete