Architecture Analysis Report

In this report, I have analyzed and proposed, compartmentally and holistically, the software architecture of my social-messaging application. It is important to note that the decisions made in this report are still subject to change, especially ones scheduled to be implemented in the later stages of development; this is due to the simple fact that many of the architecture components are complex, not to mention the app concept itself, which means new challenges, as they may appear, may require a change in the architecture. That said, let's begin our analysis.

Web-App Architecture

Frontend and Backend

Ruby on Rails is the simple choice here, as the project had been conceived as a Ruby on Rails App and much of the development is now done with the same technology. Going back now and choosing a different framework would be taking too many steps back.

Database

PostgreSQL; again, a simple choice here, as Ruby on Rails works best with PostgreSQL. Check here for listed reasons.

Mobile-App Architecture

Backend

Ruby on Rails because the backend in Ruby on Rails is most likely still usable for the mobile app.

Frontend

I have chosen React Native as the frontend framework. Kindly check the analysis below.

Frontend Framework Cost Benefit Analysis:

1) Flutter

   Cost:

   a) Flutter requires learning Dart, which is a decent learning curve.
   b) Can get laggy and harder to code as the application becomes more complex.

c) Flutter does not use native components, so the feel of the application for the user might not feel the most natural or used-to. It can still have a native look and feel; though, it may require more deliberate effort.
d) Harder to debug (asynchronous nature of the framework).

Benefits:

a) It is good for multiplatform development; it uses a single codebase and UI engine for cross-platform development, minimizing the effort required to create versions of an application for Android and iOS.
b) UI design is easier.
c) Better performance for animations and complex UI interactions.

2) React Native

Cost:

a) Performance not as good for animation and complex UI interactions vis a vis Flutter
b) Harder to debug (asynchronous nature of the framework)

Benefits:

a) Huge community for help.
b) Backed by Meta. Excellent documentation and support.
c) Great for cross-platform use, as it allows reusability of code between platforms (Android or IOS).
d) Uses Native components, feels more natural to use.
e) Want to learn React more.
f) Quick to code, therefore time to market is less with React Native compared to other frameworks.

3) Ruby on Rails

Cost:

a) For front end, other mechanisms are required to modify or fit the Rails views (basically the front end) for usage in mobile applications.

Benefits:

a) Some or much of the frontend work may be transferrable from the Web-App to the Mobile-App

4) Angular

Cost:

a) Performance not as good as React Native.
b) Decent learning curve.

c) Verbose and complex.
d) Time to market is not as good as React Native.

Benefits:

a) Backed by Google. Support tools, documentation, etc. are top notch.
b) Like React, and for the same reasons, it is great for multi-platform use.
c) Huge Community for help.
d) Uses Native Components, like React.

Decision:

I am choosing to go forward with React Native for frontend development for the following reasons:

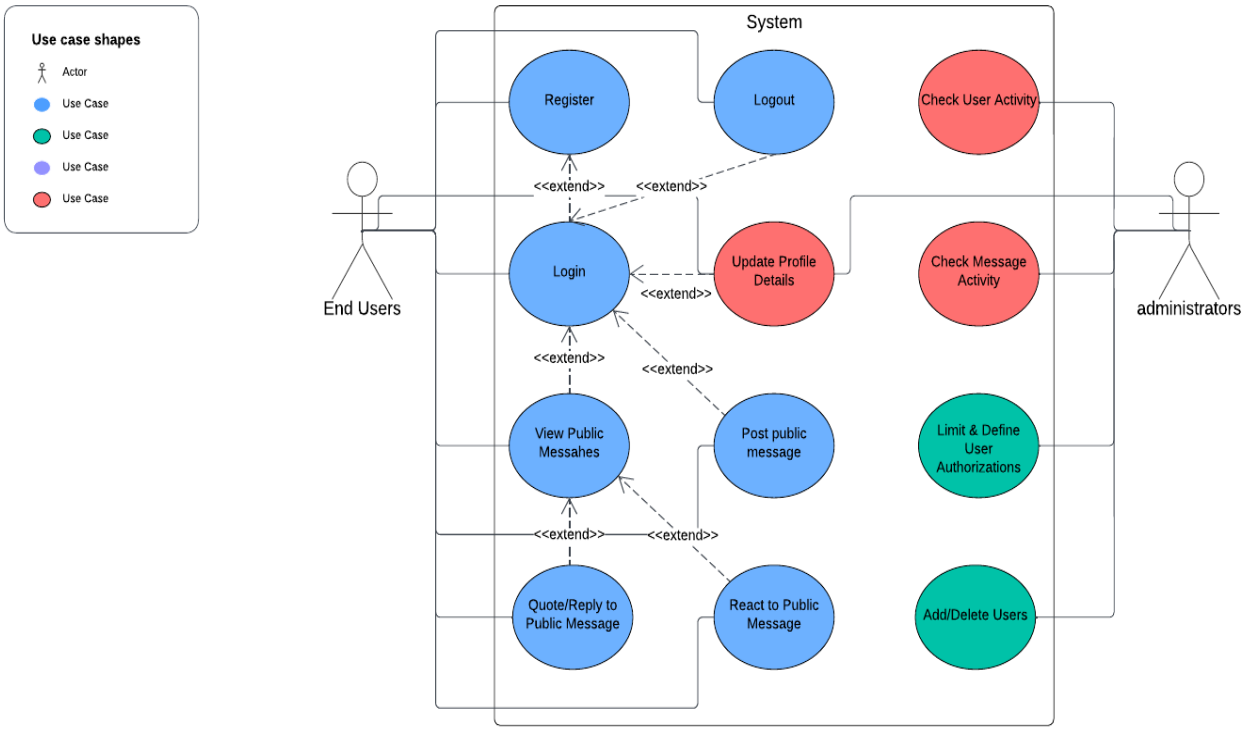
1. I wanted to learn it anyways. As far as I know, it is a great technology to have mastered for market purposes.
2. I have prior experience with React.js.
3. Time to market seems to be the least for React Native, i.e. development should be fastest with React Native.
4. The only cost-benefit balance in competition with React Native's, as per my research, is Angular, but I have no experience with Angular, and it has a decent learning curve apparently. In addition, Angular is no longer as popular.
5. It is widely supported and has a huge community for help. Therefore, there are lower chances of getting stuck with React Native.

Database

Same as Web-App.

Use Case Diagram

Following is the Use Case Diagram of my app. The diagram was made in Lucidcharts.



Deployment Diagram

Following is the deployment diagram of my app. The diagram was made in Lucidcharts.