Jamly Mobile App – Alpha Version Infrastructure Documentation

1. **General**
   1.1. The app is built upon React Native framework, version 0.48.3. It compiles and run both on Android and IOS.
   1.2. The navigation within the app is done with the package 'react-navigation'.
   1.3. UI components are used from 'native-base' and 'react-native-elements'.
   1.4. State is managed with redux and stored with 'redux-storage'.'
   1.5. API calls are done with 'redux-saga'.
2. **Structure**
   2.1. App index file (src\index.js) loads the saved redux store from disk and then renders a **SplashScreen**.
   2.2. The **SplashScreen** has two modes:
      2.2.1. If the user is logged in, it performs the first API call to get first sessions from server and shows a spinner and then renders the **App** component.
      2.2.2. Otherwise it renders the **App** component.
   2.3. The **App** component has two modes:
      2.3.1. If the user is no logged, it renders a StackNavigator with **Login** and **Signup** Components. Login and signup are currently supported only with email and password (and not social media).
      2.3.2. Otherwise, it renders a TabNavigator with 4 components (currently there is only a **Home** component).
   2.4. The **Home** component contains the following:
      2.4.1. A header with chat and record icons (not functioning yet, click on the chat icon currently logs the user out).
      2.4.2. A content container – this is where the session list should appear. You should implement this component, and a brand new SessionPage component that will be navigated from the list, with all the necessary data.
3. **State and API**
   3.1. API calls to servers are located in src\api\index.js. All the calls are done with a redux action, and the result is stored in the app state, for instance, sessions are stored in reducer called 'main' in field 'sessions'. See the reducers for more details.
   3.2. The post and get methods are defined in src\ApiUtils.js
   3.3. The src\actions\MainActions.js contains 'getSessions' and 'getFirstSessions' actions.
   3.4. The actual API calls are done in the files in the 'sagas' directory.
   3.5. Regarding the suggested API you mentioned in your estimate:
      3.5.1. The 'session' field in the state contains all the needed data for showing the session itself. You simply call in your browser (after logging) https://www.jamly.co/api/sessions?n=3, and also see it printed to the console while running the app in debug mode.

3.5.2. To get the comments, you can call /api/comments/<version id>. Notice it is called with a version (and not session) id. You can find the id of each version in the session object, in the field 'versions'. It's an array, for now you can always take the first cell, for example: session.versions[0].id. This version object also contains the data about views, likes and participants (in the 'tracks' field).

3.5.3. To publish a comment you can call (post) /api/comments/<version id> with the a 'text' field. **Please tell me before you want to try** this, because we don't want to post comments on real sessions. The api calls returns a json array of the comments, including the new one (you don't have to use it).

3.5.4. Regarding analytics, why not do it via google analytics?

3.5.5. Anyway, any problem with the API you can tell me.

4. Tasks (as defined in the estimate):

I added:

- **A strike through on the tasks I believe now unnecessary**
- **A yellow background on the tasks I believe will take now less time**
- **Comments on some**

Anyway we can discuss everything over the phone.

4.1. basic react native app ios/android, accounts, signatures, etc 1

4.2. fb login + server api authentication support 1

we already have server code to handle that. We need to adapt it for the mobile version, I suggest we wait with that a little.

4.3. ~~api wrappers and appstates: sessions, session, participants, comments, publish comment 1~~

4.4. like/share/view analytics events api wrapper 0.25

4.5. ~~screen management and routing library + header with navigation (no footer) 1.5~~

4.6. main page with infinite scroll and sessions 1

4.7. session page vertical layout 1

4.8. session page horizontal layout 1

4.9. player with vertical controls integration 1.5

I'm not sure what this is exactly. We'll discuss over the phone

4.10. player with horizontal controls integration 0.5

4.11. session participants tab 0.5

4.12. comments tab with infinite scroll 1

4.13. leave a comment (no delete or edit) with standard keyboard 1

4.14. standard share iOS/Android API 1

4.15. testing and debugging 2

4.16. android overhead 2