4 Use Cases - Useless

- Making a session
    1. On the Desktop App, you will click a button to request a session key in the Opening UI
    2. On the Desktop App, a call will be made to the App Data Module
    3. On the Desktop App, the call will be forwarded to the Network Module
    4. On the Server, the request will be received via an http connection.
    5. On the Server, the key will be generated and saved
    6. On the Desktop App, the key will be received
    7. On the Desktop App, the key will be sent back to the Opening UI, saved, and displayed.
- Joining a session
    1. On the Android App, the user will enter in the session key that was displayed
    2. On the Android App, a call will be made to the network service.
    3. On the Server, the request will be received by an http connection
    4. On the Server, the Socket.io room will be found using the session key
    5. On the Server, the user will join the Socket.io room.
    6. On the Server, a response will be sent to the Android App telling it whether or not it connected.
    7. On the Android App, if a successful response is received, an Intent will be sent to move to the Answer Questions UI.
    8. Otherwise, the Android App will display an error message notifying the user of a failed connection attempt.
- Asking a Question
    1. On the Desktop App, the user will create questions and answers
    2. On the Desktop App, the question and answers will be sent to the server via json over http
    3. On the Server, the question, answers, and the session key will be received.
    4. On the Server, the information that was received will be sent to the Server Data module to be saved via json.
    5. On the Server, a session will become live and push the question and the possible answers to all connected android users.
    6. The server will wait for android users to answer and for new android users to connect.
- Receiving the Answers
    1. On the Desktop App, the user will make a call to close the session.
    2. On the Desktop App, the call will be forwarded to the App Data Module.
    3. On the Desktop App, the call will be forwarded to the Network Module.
    4. On the Server, the request will be received via http connection.
    5. On the Server, the session manager will make a call to collect from Server Data.
    6. At the same time the Android Network Module will receive a call from the server Session manager to put the UI in a standby state.

7. On the Android App, the network module will make a call to Ui to make it standby.
8. At the same time, on the Server, the session manager will send the data back to the desktop module via http.
9. On the Desktop app, the network manager will forward the results to the data manager.
10. On the Desktop App, the data manager will forward the data to the display results UI.