Vorab:

- Dieses Dokument ist adressiert an Gruppe 7 und wurde nach einer Absprache von Gruppe 3 erstellt.
- Unser (Gruppe 3) UI wird ähnlich aussehen und funktionieren wie WhatsApp Web.
- Wir haben angenommen, dass alle untenstehenden Methoden von euch (Gruppe 7) implementiert werden. Falls etwas nichts mit euch zu tun hat sondern mit einer anderen Gruppe, so teilt uns dies bitte mit.
- Wir sollten eigentlich mit diesen 5 Methoden alles abgedeckt haben. Falls das nicht der Fall ist, würden wir euch dies selbstverständlich frühestmöglich mitteilen damit genügend Zeit bleibt diese zu implementieren.

Schnittstellen die wir gerne hätten

1. **Login**:

   Ausgangssituation:
   Benutzer hat bereits ein Benutzerprofil in der Datenbank und möchte sich (z.B. auf einem neuen Gerät) anmelden.

   Vorschlag:
   Boolean loginAttempt(String Benutzername, String Passwort[1]) {...}

   Diese Methode wird vom Benutzer aufgerufen wenn er um Erlaubnis fragt, auf seine Datenbank zuzugreifen *(= alle von ihm gesendeten Nachrichten und an ihn adressierten Nachrichten. Wahrscheinlich werden wir es noch erweitern auf Bilder[1] und PDFs, die man versenden kann)*. Er meldet sich an mit seinem *Benutzernamen* und seinem *Passwort*. Der Rückgabewert könnte ein Boolean sein *(True für Zugriff gestattet False für Zugriff verweigert.)*.

   [1] Eventuell ist ein String hier ungünstig da nicht verschlüsselt.
   [2] Diese werden voraussichtlich als base64 String gespeichert, sprich als String übergeben mit einem Identifier(?). Alternativ zu einem Identifier könnte man auch 3 Einträge im Array haben bei denen jeweils nur eines gefüllt ist

2. **Registrieren**:

   Ausgangssituation:
   Der Benutzer hat noch kein “Konto” und möchte dem Netzwerk beitreten. Er wählt seinen *Benutzernamen* und *Passwort*.

   Vorschlag:
   Boolean registrationAttempt(String Benutzername, String Passwort) {...}

   Die Methode soll *(in ähnlicher Weise wie (1.))* es dem Benutzer erlauben, sich dem Netzwerk anzuschliessen. Die Registrierung mittels Benutzername und Passwort ist ein Vorschlag, da wir nicht genau wissen, wie genau das Onboarding funktionieren wird und welche Daten der Benutzer dazu bereitstellen soll (?). Auch hier: Wenn das Passwort verschlüsselt werden soll mit einer Methode, so teilt uns dies bitte mit, dann machen wir das. (Es muss ja auch noch überprüft werden, ob der Benutzername bereits verwendet wird z.B.)

3. **Bestehende Nachrichten ausgeben** (welche an den Nutzer adressiert sind):

   Ausgangssituation:
   Der Nutzer hat die App soeben gestartet und seine Nachrichten müssen neu geladen werden.

   Vorschlag:
   *String**[]** returnAll() {...}*

   Diese Methode soll die Datenbank noch nicht automatisch aktualisieren, sondern nur die bereits in der Datenbank vorhandenen Daten als *Array* ausgeben. Am liebsten wäre es uns, wenn das Array aus dictionaries oder tuples oder weiteren Arrays besteht, damit wir einfach zugreifen können auf; die ***Zeit*** *(wann es versendet wurde)* */ den* ***Sender*** *(von wem es kommt)* */ den* ***Empfänger*** *(an wen die Nachricht geht [(3)])* ***Kontext*** *(ob es sich um eine alleinstehende Nachricht (Kontext self) oder um eine Antwort auf eine andere Nachricht (Kontext = ID der anderen Nachricht))* */ die* ***Nachricht*** *(was wurde geschrieben)* */* ***ID*** *(Eindeutige ID der Nachricht)* */ etc.*[(4)]

   Bemerkung: Wir haben noch nicht ganz ganz bedacht, wie genau das topological sorting ins Spiel kommen kann. Auf jeden Fall soll es hier (3.) bereits topologisch sortiert wiedergegeben werden.

   [(3)] Wir haben es so verstanden, dass jede Person eine eigene Datenbank hat, in der alle Daten gespeichert sind, welche den Benutzer selbst betreffen. Deshalb sollte der Empfänger hier immer der Benutzer selbst sein ausser wenn es sich um ausgehende Nachrichten handelt.
   [(4)] Bemerkung: siehe pcap-file-Struktur in Ersatz-Datenbank von Herrn Tschudin.

4. **Neue Nachrichten ausgeben** (welche an den Nutzer adressiert sind):

   Ausgangssituation:
   Der Nutzer war eine Zeit lang offline, hat jetzt eine Internet Verbindung und hat inzwischen neue Nachrichten erhalten. Er klickt auf “update” um die neuen Nachrichten anzeigen zu lassen.

   Vorschlag:
   *String**[]** returnNew() {...}*

   Diese Methode soll 2 Sachen machen:
   1. Sie soll alle neuen (noch nicht vorhandenen) Nachrichten der Datenbank hinzufügen.
   2. Sie soll diese neu erhaltenen Nachrichten als Array ausgeben (wie bei (2.)).

   Bemerkung: Irgendwie müssen wir wissen, an welchen Stellen die Nachrichten eingefügt werden müssen (topological sort). Wir haben gleich wie bei (3.) noch nicht darüber nachgedacht, wie genau das passieren kann (?). Vielleicht hat jede Nachricht zusätzlich eine Nummer, die deren relativen Platz innerhalb eines Chats angibt. Wenn diese Methode aufgerufen wird könnte man jene Nachrichten neu senden, dessen Nummer sich geändert hat. Wir erkennen dann, dass die ID bereits vorhanden ist und es sich also um eine Änderung handelt, löschen dann jenen veralteten Eintrag und setzen die Nachricht am richtigen Ort (according to the given number) neu ein.

5. Zu sendende **Nachricht speichern**:

   Ausgangssituation:

Eine Nachricht wird erfasst und per “send” versendet. Gegeben: Empfänger der Nachricht und Inhalt der Nachricht.

Vorschlag:
void saveMessage(String Receiver, String Nachricht) {...}

Dieser Methode soll der *Receiver* der Nachricht (entweder eindeutiger Benutzername/Gruppenchat Name oder Benutzer ID/ Gruppenchat ID. Falls letzteres brauchen wir noch eine getter Methode um diese ID zu erhalten.) und die zu sendende *Nachricht* übergeben werden. Der Nachricht muss dann wie bei (3.) folgendes zugewiesen werden; Zeit, Sender, **Empfänger** *(an wen es geschickt werden soll)*, Kontext, Nachricht und ID. Die Nachricht soll dann in der Datenbank abgelegt werden in dem es an Betreffende Personen geschickt wird.