Da ich und Walid im selben Haus wohnen , treffen wir uns regelmässig (jede Woche) und pairprogrammieren die Aufgaben.

Protoll von Woche 1
-Rust installieren
-Datenbank ChronicleDB zum laufen bringen, damit man damit arbeiten kann (stundenlang)
-c++ Komponenten installiert für ChronicleDB
-Entscheidung für react-framework für Javascript
-Repository erstellt und ersten commit gepusht für Frontend in react js
-Für react js wird noch nodejs benötigt, installiert
-Update: react Installation erwies sich bei Walid als nicht so einfach und hat
Probleme.
Frontend: -Ein paar Buttons hinzugefügt mit teilweise Funktionalität
-Ein Button kann eine Anfrage für dummy Daten starten (zum Testen)
-Vertraut gemacht mit react js Bibliothek bzw. mit der Javascript Syntax, also die ersten Schritte
-ESLint (live compiler für Format und direkte Fehleranalyse)

Protokoll von Woche 2
- Mina hat Probleme die ChronicleDB zum Laufen zu bringen ,da die mit der neuen M-Archtikitur von Apple nicht kompatible ist
Eine Lösung hat er, in dem er die ChronicleDB auf eine Linuxmaschine im selben netzwerk laufen lässt.

- Feuern der möglichen Requests in Postman , damit wir uns mit dem Format der Response und vertraut machen.
und die Collection als JSON zwischen uns exportet.

Protokoll von Woche 3
- Zusammenbau vom Create_Stream_Request in Axios (eine Bibliothek von JS , mit der man HTTP-Requests schicken kann ) , allerdings kommt immer ein NetzwerkFehler

Protokoll von Woche 4
-ein neues Build von ChronicleDB von Amir bekommen , wo dieser Netzwerkfehler nicht mehr auftritt .
- wir haben uns bei Walid getroffen , und die Tasks pairprogrammiert.
- den create_Stream_implementieren
-den Show_stream_implementieren

Protokoll vom 12.12.2021
- den Rest der get_Requests implementieren.
- Walid hat das Design etwas aufgehübscht.

Protokoll vom 18.12.2021
- Thema Authentication and Authentifizierung:
Mina übernimmt das Thema , in dem er eine NodeJS RestApi schreibt , die die User Details in eine PostGr Tabelle schreibt
wo auch die Passwörter gehasht sind

Protokoll von 29.12.2021
-Walid arbeitet an den LoginScreen und RegistrierungScreen
- Mina macht an dem Authentication weiter.

Protokoll von 06.01.2022
- jetzt ist das Backend Authentication erfolgreich umgesetzt
ein eigenes privates Repo dafür erstellt und Amir dazu eingeladen
Repo can be found here
https://github.com/MinaTheDebugger/Authentication_BackEnd
- es ist so modular wie möglich umgesetzt
-Der Velauf der Authentication :
-eine Postgress Connection wird gebaut
-ein Post request gegen /register mit den usersdetails schicken
wird in eine Tabelle gespeichert
beim Login wird ein PostRequest gegen /login mit dem Email und Pass schicken
wenn 200 dann wird der angemeldete User zum HauptScreen zugelassen.
wenn 4xx , dann wird der user nicht zugelassen.

Protokoll von 15.01.2022
- wir haben ein Problem mit dem SystemLoad Request
Amir hat uns ein neueres Build geschickt , allerdings geht die bei uns in der lokalen Entwicklung nicht , wahrscheinlich Konflikte der Versionen von Rustc und clang
das müssen wir uns noch weiterschauen

Protokoll von 22.01.2022
- wir überlegen uns noch , wie man einem bestimmten User nur seine eigene Streams anzeigen kann .
momentan ist es so man ist entweder angemeldet und darf alles (Streams

erzeugen , queryTimetravel , show , etc) oder man ist nicht angemeldet und hat keinen Zugriff auf die App .