Team Working Agreement

Cartrekk

Mon, Feb 24

**Logistics**

* Work Room: Living room with Sahil on Discord
* Meetings times
  + Monday 1:00 PM
  + Tuesday 9:50 AM (TA)
  + Wednesday 8:00 PM
  + Friday 7:00 PM
* Project repository
* Code Style: Swift Style Guide
* Repository: <https://github.com/ameekals/Cartrekk>
* Primary communication: Discord
* Dev env: Xcode
  + Platform: iOS/macOS

**Acceptance Criteria**

**User Story: As a driver, I want to equip a car, make it show up on the map, and add it to my routes. Points: 5**

## **Task 3.1: Add cars to routes for other users**

* Car information saved with route data
* Other users can see which car was used on shared routes
* Car information displays in route details
* Car data syncs properly with backend
* Handles scenarios where car data is unavailable

## **Task 3.2: Add car equipment functionality**

* User can select from available cars in garage
* Selected car is marked as "equipped"
* Only one car can be equipped at a time
* Equipment status persists across app sessions
* User receives confirmation when car is equipped
* UI clearly shows which car is currently equipped

## **Task 3.3: Replace default blue location tracker with user's 3D car**

* Custom 3D car model replaces standard location marker
* Car model faces correct direction of travel
* Car model is properly scaled on map
* Smooth animation during movement
* Car model loads efficiently without performance issues

**As a music listener I want Spotify integration so I can log which songs I listened to during a route. Points: 5**

## **Task 2.1: Setup Spotify login connectivity**

* Spotify authentication flow implemented
* User can connect/disconnect Spotify account
* Authentication tokens stored securely
* Handles authentication errors gracefully
* Login status persists across app sessions
* Clear indication of connection status

## **Task 2.2: Track Spotify songs during route**

* Current playing song captured during route recording
* Song data (title, artist, album) saved with route data
* Song history displayed in route details view
* Timestamps for songs align with route segments
* Handles offline/disconnected scenarios
* Interface shows current song during active route
* Syncs song data with backend storage