HUD Customization Feature – Era Smart Glasses

This project demonstrates a customizable heads-up display (HUD) for smart cycling glasses, allowing users to control which performance metrics appear during a ride.

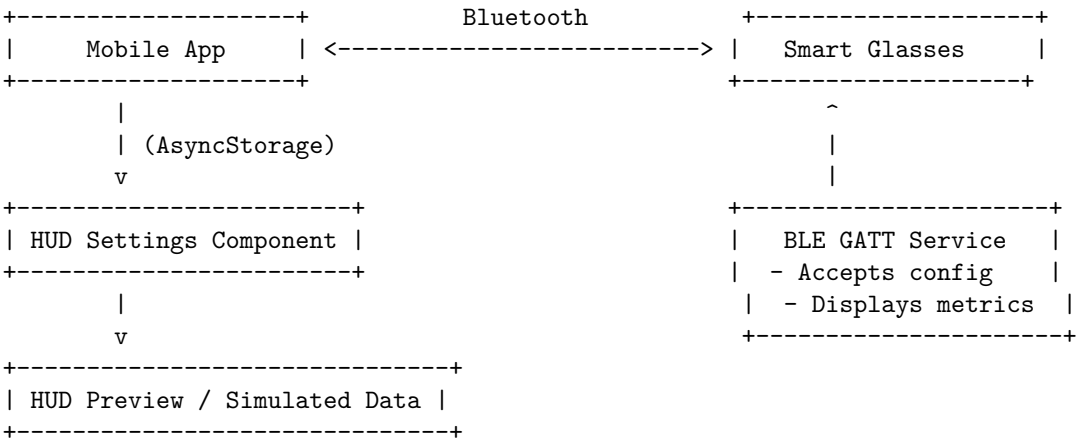
The core idea is to let cyclists use a mobile app to select the top three metrics they want to see (e.g., speed, heart rate, cadence). These preferences are stored locally and transmitted via Bluetooth Low Energy (BLE) to the glasses, which render the chosen data in real-time.

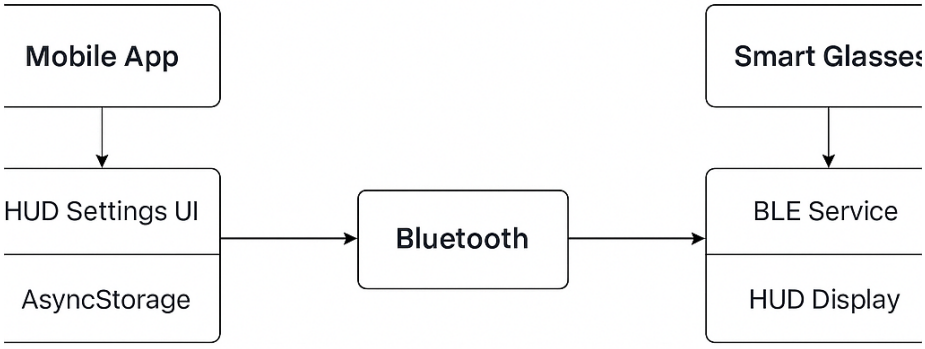
Why This Project

Customizability enhances user experience. Instead of showing the same fixed stats for everyone, this feature gives each rider the ability to define what's most important to them. The UI was built with simplicity in mind—easy metric selection and a preview panel for reassurance before riding.

Repo: https://github.com/ryannnsevidal/customizable-hud

System Architecture





Feature Workflow

1. **Initialization**
   The user launches the app and navigates to the HUD settings page. A list of metrics is displayed.

2. **User Selection**
   Up to three metrics can be selected. A HUD preview updates in real-time to reflect choices.

3. **Configuration Storage**
   User choices are saved using `AsyncStorage` in this format:

   ```
   {
     "hudMetrics": ["speed", "heart_rate", "cadence"]
   }
   ```

4. **Bluetooth Sync**
   Once the smart glasses are powered and connected, the app:

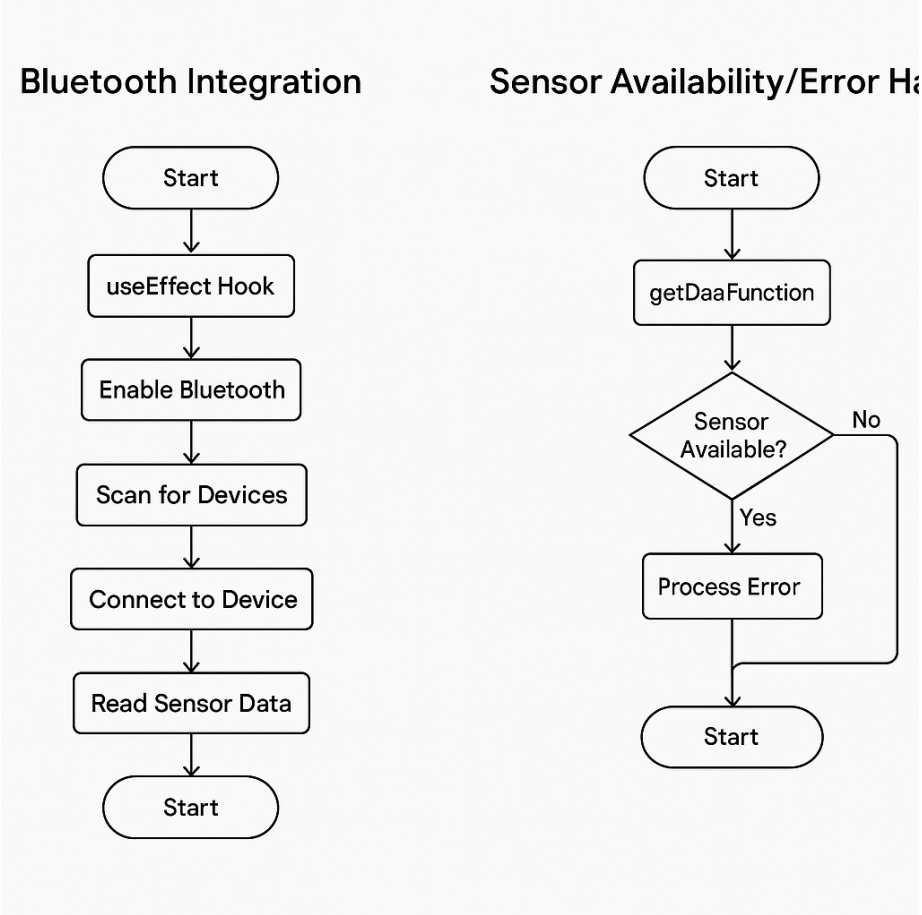


Figure 1: Blue

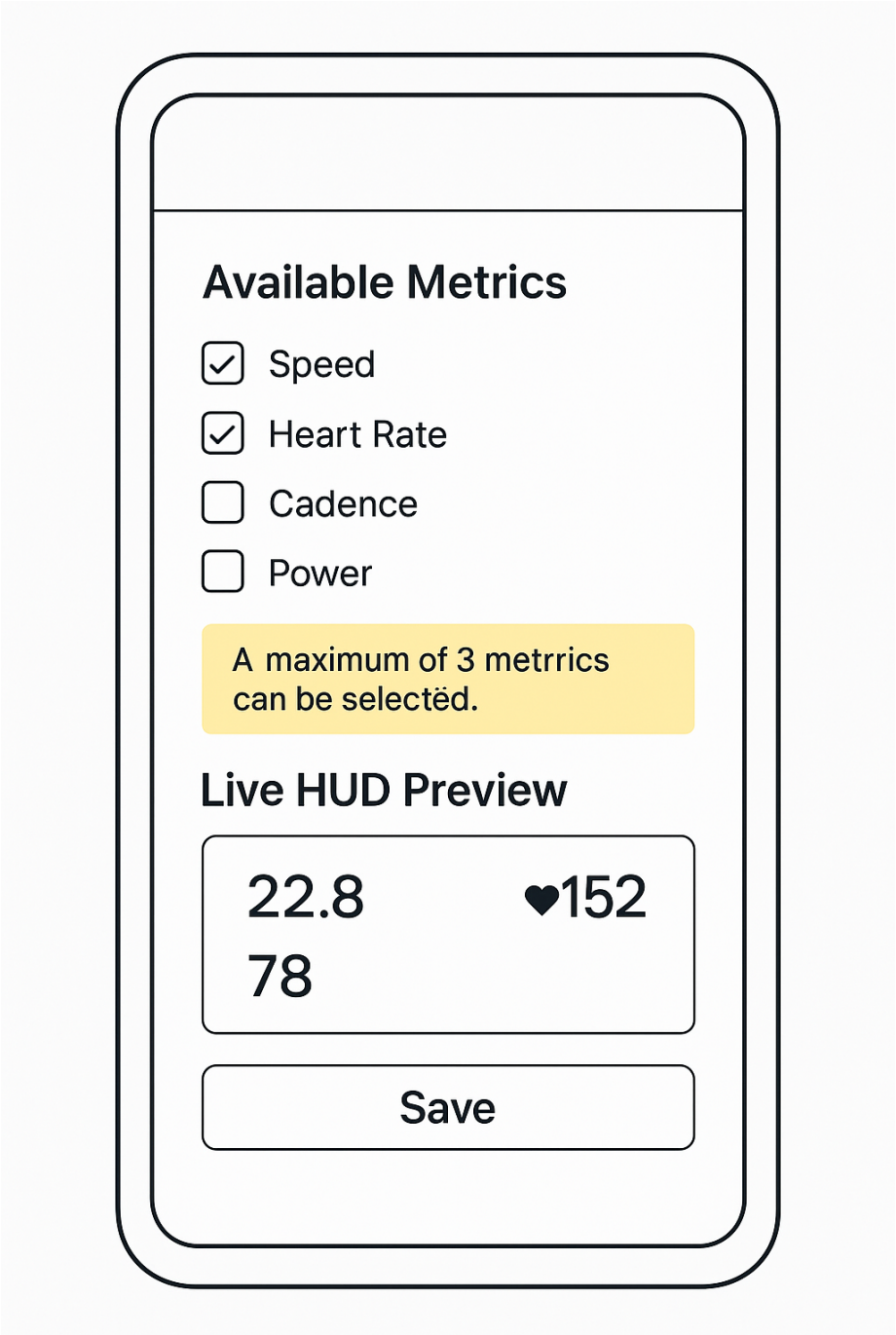
- Scans and connects via BLE
- Discovers GATT services
- Writes the selected metrics to a writable characteristic

5. **Ride Time**
   During the ride, the app streams real-time data to the glasses. Only the selected metrics are shown.

6. **Handling Edge Cases**
   If a selected metric isn't available (e.g., power meter is missing), it is substituted with a default. If the Bluetooth connection drops, the last known config is preserved or an alert is issued.

HUD Preview Example

```
+--------------------------+
| SPEED: 22.5 km/h         |
| HEART RATE: 145 bpm      |
| CADENCE: 88 rpm          |
+--------------------------+
```



Bluetooth Implementation

This project uses `react-native-ble-plx` for BLE functionality.

Scan and Connect

```
const manager = new BleManager();

manager.onStateChange((state) => {
  if (state === 'PoweredOn') {
    manager.startDeviceScan(null, null, (error, device) => {
      if (device.name?.includes('EraGlasses')) {
        manager.stopDeviceScan();
        device.connect().then((d) => d.discoverAllServicesAndCharacteristics());
      }
    });
  }
});
```

Transmit HUD Configuration

```
const configJSON = JSON.stringify({ hudMetrics: ["speed", "heart_rate", "cadence"] });
const encoded = Buffer.from(configJSON).toString('base64');

await device.writeCharacteristicWithResponseForService(
  'SERVICE_UUID',
  'CHARACTERISTIC_UUID',
  encoded
);
```

Sensor Availability Check

Before sending configuration to the glasses, the app checks whether selected metrics are supported by available sensors.

Example logic:

```
const isSensorAvailable = (sensorType) =>
  connectedDevices.some((device) => device.name.includes(sensorType));

const validatedMetrics = selectedMetrics.filter(isSensorAvailable);
```

Simulated Data (for Preview Testing)

To demonstrate HUD updates without real sensors, random values are generated:

```
useEffect(() => {
  const interval = setInterval(() => {
    setPreviewData({
      speed: (Math.random() * 40).toFixed(1),
      heart_rate: Math.floor(Math.random() * 70 + 100),
      cadence: Math.floor(Math.random() * 90 + 60),
```

```
    });
  }, 1000);
  return () => clearInterval(interval);
}, []);
```

Tech Stack and Decisions

Layer	Technology	Reason
Frontend	React Native	Cross-platform, supports BLE
Storage	AsyncStorage	Lightweight, persistent on-device
Bluetooth API	react-native-ble-plx	Active maintenance, broad device support
Format	JSON	Readable and easy to encode/decode

Next Steps / In Progress

- BLE UUIDs to be finalized using the Era SDK or GATT explorer
- Real sensor data integration
- Additional error handling for mid-ride disconnects

For Reviewers

This repo contains the front-end and preview logic for HUD customization. BLE connection and configuration sync logic is laid out and ready for integration. The UX is structured to be intuitive and clean, especially for riders adjusting settings on the go.

If you'd like visuals added or this turned into a demo-ready PDF, feel free to request it.