**AuraDrive: MVP Execution Plan**

This plan is designed for a rapid, iterative development cycle (like a hackathon) to build a functional MVP that impresses evaluators. It leverages your existing progress on BLE mesh and focuses on parallel work streams.

**Team Composition:**

* **Mobile Team (2 Developers):** Focus on the Android/Flutter app, data acquisition, and user interface.
* **Backend Team (2 Engineers):** Focus on the networking layer's reliability, data flow, and serverless architecture.
* **ML Team (2 Specialists):** Focus on sensor fusion, positioning algorithms, and collision logic.

**Phase 1: Data Pipeline & Network Validation (First 12 Hours)**

**Objective:** Establish a stable flow of raw sensor data from the mobile app, through the validated network, and into a log file for the ML team.

* **Lead Team:** **Mobile** | **Supporting Team:** **Backend**
* **Action Description:**
  1. **Mobile Team:** Develop the core Flutter service to access and continuously capture raw GNSS (pseudoranges, doppler, etc.) and IMU (accelerometer, gyroscope) data from the phone's hardware.
  2. **Backend Team:** Work with the Mobile team to define a rigid JSON data packet structure. This packet will contain sensor data, timestamps, and a unique device ID.
  3. **Joint Task:** Integrate the existing BLE mesh module into the Flutter app. The Mobile team will handle the frontend integration, while the Backend team will ensure the data packets are correctly serialized and broadcasted over the mesh network.
* **"How" Details (Key Tech/Process):**
  1. **Mobile:** Flutter plugins for sensor access (geolocator, sensors\_plus, custom platform channels for raw GNSS).
  2. **Backend:** Formalize a JSON schema. Validate the existing BLE mesh for packet loss and latency with the defined data packets.
  3. **Handoff:** The Mobile team provides the Backend team with a simple app build for network testing. At the end of this phase, a CSV or text log of raw data from multiple devices must be handed off to the ML team.

**Phase 2: Sensor Fusion & Collision Logic Prototyping (Parallel to Phase 1)**

**Objective:** Develop the core mathematical models for positioning and collision detection. This phase runs in parallel using pre-existing or quickly-gathered sample data.

* **Lead Team:** **ML**
* **Action Description:**
  1. **EKF Development:** Using the data logs from Phase 1 (or sample data initially), the ML team will prototype the Extended Kalman Filter (EKF) in a Python environment. The goal is to fuse the noisy GNSS and IMU data to produce a more stable and accurate position/velocity estimate.
  2. **Collision Logic:** Design the initial collision prediction algorithm. This will start as a simple distance-and-velocity vector calculation (predicting future positions) but should include logic for the "Adaptive Safety Envelope" – making the warning zone larger or smaller based on the EKF's position confidence score.
* **"How" Details (Key Tech/Process):**
  1. **ML:** Python, NumPy, SciPy. Develop in a Jupyter Notebook or similar environment for rapid iteration. Test against static and simple motion scenarios.
  2. **Input:** Requires the sensor data logs from the Mobile team as soon as they are available.

**Phase 3: On-Device Implementation & UI Mockups (First 24-36 Hours)**

**Objective:** Port the prototyped algorithms to the mobile device and build the user-facing interface.

* **Lead Teams:** **ML** & **Mobile** | **Supporting Team:** **Backend**
* **Action Description:**
  1. **ML Team:** Begin porting the validated Python EKF and collision logic to Dart/Kotlin. The goal is an efficient, on-device library that can process the real-time data stream.
  2. **Mobile Team:** Develop the main UI/UX for the application. This includes a simple visualization of nearby peers (dots on a screen), clear visual indicators for proximity warnings (e.g., green/yellow/red auras), and a debug view showing key metrics (position confidence, peer count).
  3. **Backend Team:** Refine the peer-to-peer networking layer. Implement a seamless switch between BLE Mesh (for low-power discovery) and Wi-Fi Direct (for higher-bandwidth data exchange when peers are close), which is a key innovative enhancement.
* **"How" Details (Key Tech/Process):**
  1. **ML:** Translate NumPy logic into Dart. Focus on performance to avoid draining the battery.
  2. **Mobile:** Flutter for UI. Design should be clean and distraction-free, prioritizing the alert system.
  3. **Backend:** Implement a service discovery protocol that hands off connections between BLE and Wi-Fi Direct based on signal strength or data requirements.

**Phase 4: Full System Integration & Testing (36-60 Hours)**

**Objective:** Combine all components into a single, functional application and conduct rigorous real-world testing.

* **Lead Team:** **ALL**
* **Action Description:**
  1. **API Handoffs:** The ML team provides a clean API for the Mobile team to get fused position data and collision alerts. The Backend team provides a simple API to send/receive data over the hybrid network.
  2. **Integration:** The Mobile team integrates the ML library and the refined networking module into the Flutter application. The app should now be able to: capture data, share it with peers, receive peer data, process it through the EKF, and display alerts.
  3. **Field Testing:** All team members participate in field tests.
     + **Scenario 1 (Static):** Place phones at known distances to calibrate the EKF.
     + **Scenario 2 (Dynamic):** Walk/drive with phones to test real-time alerts and trajectory predictions.
     + **Scenario 3 (Indoor Demo):** Use static phones as anchor beacons to test indoor navigation capabilities.
* **"How" Details (Key Tech/Process):**
  1. **Collaboration:** Use Git for version control with a clear branching strategy (e.g., feature/networking, feature/fusion-engine).
  2. **Debugging:** Use adb logcat and Flutter DevTools to monitor app performance, data flow, and identify bugs in real-time.

**Phase 5: Polish, Presentation & Metrics (Final 12 Hours)**

**Objective:** Finalize the demo, gather performance metrics, and prepare the presentation.

* **Lead Team:** **ALL**
* **Action Description:**
  1. **Bug Fixing & UI Polish:** Based on field testing, fix critical bugs and refine the UI for clarity and impact.
  2. **Metric Collection:** Quantify the system's performance. Measure the positioning accuracy (vs. a high-precision GPS), the latency of alerts, and the maximum number of peers supported on the network.
  3. **Demo & Pitch Prep:** Record a smooth video of the app working in a real-world scenario. Prepare a slide deck that emphasizes the key innovations: decentralization, low cost, adaptive safety, and the impressive quantitative metrics you collected.
* **"How" Details (Key Tech/Process):**
  1. **Framing for Evaluators:** Focus the pitch on how you solved a high-cost problem with existing hardware, emphasizing the novelty of the serverless fusion and hybrid networking. Be ready to show your modular and well-documented code.