DebrisScan AI - Roles

👥 TEAM STRUCTURE

- **Engineer 1 → AI / ML Engineer (Core Intelligence)**
- **Engineer 2 → Backend Engineer A (Data & AI Services)**
- **Engineer 3 → Backend Engineer B (SSA Logic & Integration)**
- **Engineer 4 → UI / Frontend Engineer (Visualization & Demo)**

🧠 ENGINEER 1 — AI / ML ENGINEER

("Brain of the system")

Goal

Build, train, and evaluate **all AI models**.

Step-by-Step Tasks

🔹 Phase 1: Dataset Understanding (Week 1)

1. Download **all 4 datasets** (A–D)
2. Inspect formats (images, CSV, `.npy`)
3. Create preprocessing scripts for:
    - Images (resize, normalize)
    - TLE → state vectors
    - Radar heatmaps

🔹 Phase 2: Model Development (Week 2)

1. Train **YOLOv8** on debris image dataset
2. Fine-tune YOLOv8 on **optical streak dataset**
3. Build **orbit prediction model**
    - Input: (x,y,z,vx,vy,vz) sequences
    - Model: LSTM / Transformer
4. Train **radar feature encoder (CNN)**

🔹 Phase 3: Fusion & Evaluation (Week 3)

1. Combine outputs:
    - Vision confidence
    - Orbit uncertainty
    - Radar features
2. Generate **collision risk score**
3. Evaluate:
    - mAP (vision)
    - RMSE (orbit)
    - False alarm rate

Deliverables

✅ Trained models
✅ Evaluation plots
✅ Model inference scripts
✅ Results for paper

⚙️ ENGINEER 2 — BACKEND ENGINEER A

("AI Pipeline & APIs")

Goal

Expose AI models as **clean backend services**.

Step-by-Step Tasks

🔹 Phase 1: Backend Setup (Week 1)

1. Setup **FastAPI**
2. Define API contracts:
    - `/detect-image`
    - `/predict-orbit`
    - `/radar-features`

🔹 Phase 2: Model Integration (Week 2)

1. Load AI models from Engineer 1
2. Build inference pipelines
3. Handle:
    - Input validation

- Batch requests
- Latency optimization

- **Phase 3: Storage & Logging (Week 3)**

1. Store results (detections, risk scores)
2. Log inference metadata
3. Prepare JSON responses for frontend

Deliverables

✅ AI inference APIs
✅ Clean request/response schemas
✅ Backend documentation

🛰️ ENGINEER 3 — BACKEND ENGINEER B

("SSA Logic & System Intelligence")

Goal

Implement **space-specific logic**, not ML.

Step-by-Step Tasks

- **Phase 1: Orbital Mechanics (Week 1)**

1. Implement **SGP4 orbit propagation**
2. Parse CelesTrak GP CSV
3. Generate future trajectories

- **Phase 2: SSA Logic (Week 2)**

1. Compute:
   - Distance between objects
   - Time to closest approach
2. Compare:
   - AI prediction vs classical SGP4

🔹 **Phase 3: Risk Assessment (Week 3)**

1. Define collision thresholds
2. Generate **risk levels**:
    - Low / Medium / High
3. Send alerts to backend APIs

Deliverables

✅ Orbit propagation engine
✅ Collision logic module
✅ Risk assessment service

🎨 **ENGINEER 4 — UI / FRONTEND ENGINEER**

("Face of the project")

Goal

Build a **clean, demo-ready interface**.

Step-by-Step Tasks

🔹 **Phase 1: UI Planning (Week 1)**

1. Design dashboard layout:
    - Image detection panel
    - Orbit plot panel
    - Risk score panel
2. Choose stack:
    - React / Next.js
    - Chart.js / Three.js

🔹 **Phase 2: UI Development (Week 2)**

1. Integrate backend APIs
2. Display:
    - Bounding boxes on images
    - Orbit trajectories (2D/3D)
    - Risk alerts

🔹 **Phase 3: Polish & Demo (Week 3)**

1. Improve UX
2. Add:
    - Status indicators
    - Explanation tooltips
3. Prepare **demo flow**

Deliverables

✅ Functional dashboard
✅ Clear visualizations
✅ Demo-ready UI

🧩 HOW EVERYTHING CONNECTS

```
Sensors/Datasets
      ↓
AI Engineer → Models
      ↓
Backend A → APIs
      ↓
Backend B → SSA Logic
      ↓
Frontend → Dashboard
```

🗓️ FINAL WEEK (ALL ENGINEERS)

Week 4 (Feb last week)

- Integrate all components
- Run final experiments
- Generate figures
- Write paper + project report
- Record demo (if needed)

✅ ONE-LINE TEAM SUMMARY (FOR REVIEW / PPT)

> *"The system was developed by a four-member team with dedicated roles in AI modeling, backend intelligence, SSA logic, and frontend visualization, enabling an end-to-end AI-powered space debris detection and tracking platform."*