ERES PlayNAC “KERNEL” Codebase Version 7.1

New Age Cybernetic Game Theory Core Codebase

Preface: This update refines v7.0 using the Claude.ai recommendations, focusing the kernel on a human-verified skill development platform and eliminating legacy complexity. Key changes:

- EarnedPath is now a binary skill-credential engine
- BioPoW replaced by BiometricAuth for proof-of-human
- 17×7 Keyword Matrix reduced to 7 core development areas
- GAIA becomes an ExpertAdvisor advisory module
- JASConsensus → PeerReviewEngine for collaborative validation
- MediaProcessor streamlined for creative feedback
- Removed: GiantERP, token minting, complex formulas, mystical elements

---

📦 Revised Repository Structure

```
playnac-kernel-v7.1/
├── docs/
│   ├── overview.md          # Vision & architecture diagrams
│   └── core-areas.md        # 7 Core Development Areas definitions
├── src/
│   ├── kernel/
│   │   ├── config.py        # Env loader & validator
│   │   ├── context.py       # Session/context manager
│   │   ├── storage.py       # Persistence adapter (SQLite/Postgres)
│   │   └── kernel.py        # Orchestrator (ingest → modules)
│   ├── earnedpath/
│   │   └── ep_engine.py     # Binary skill progression & credentialing
│   ├── auth/
│   │   └── biometric.py     # BiometricAuth: heartbeat/voice check
│   ├── advisory/
│   │   └── expert_advisor.py  # ExpertAdvisor: curriculum guidance
│   ├── review/
│   │   └── peer_review.py   # PeerReviewEngine: project validation
│   ├── media/
│   │   └── processor.py     # MediaProcessor: feedback engine
│   ├── intent/
```

```
│   │   └── intent_parser.py   # Maps text to intent + core area
│   ├── utils/
│   │   ├── exceptions.py      # KernelError, etc.
│   │   └── logger.py          # Structured JSON logging
├── tests/                     # Unit & integration tests
├── examples/                  # Demo scripts & YAML configs
├── .github/                   # CI workflows (lint, test)
├── Dockerfile                 # Container spec
├── .env.example               # Env vars template
├── pyproject.toml             # Python deps & entry points
├── requirements.txt           # Python dependencies
├── README.md                  # Overview & usage
├── LICENSE                    # MIT License
└── CHANGELOG.md               # v7.0 → v7.1 history
```

🛠️ Detailed Module Stubs

These stubs implement the simplified kernel; fill in logic per docs/architecture.

src/earnedpath/ep_engine.py

```
class EarnedPathEngine:
    """
    Binary skill progression & credential issuance.
    """
    def __init__(self):
        self.skills: dict[str, bool] = {}

    def unlock_skill(self, skill_id: str, prereqs: list[str]) -> bool:
        if all(self.skills.get(p, False) for p in prereqs):
            self.skills[skill_id] = True
            self.issue_credential(skill_id)
            return True
        return False

    def issue_credential(self, skill_id: str) -> None:
        # Persist to storage & emit event
        pass
```

src/auth/biometric.py

```
class BiometricAuth:
    """
    Simple proof-of-human: heartbeat or voice pattern check.
    """
    def verify(self, sample: bytes) -> bool:
        # Stub: validate heartbeat or voice signature
        return True
```

src/advisory/expert_advisor.py

```
class ExpertAdvisor:
    """
    AdvisoryClient for expert curricular recommendations.
    """
    def __init__(self, advisors: list[str]):
        self.advisors = advisors

    def recommend(self, area: str) -> list[str]:
        # Return curriculum modules per advisor consensus
        return []
```

src/review/peer_review.py

```
class PeerReviewEngine:
    """
    Community-driven project validation.
    """
    def submit(self, project_id: str, user_id: str) -> None:
        pass

    def vote(self, project_id: str, reviewer_id: str, score: int) -> None:
        pass

    def rating(self, project_id: str) -> float:
        return 0.0
```

src/media/processor.py

```
class MediaProcessor:
    """
    Provides automated feedback on creative submissions.
    """
    def assess(self, data: Any) -> dict:
        # Return {'quality': float, 'feedback': str}
        return {'quality': 0.0, 'feedback': ''}
```

src/intent/intent_parser.py

```
CORE_AREAS = [
    'Technical Skills', 'Communication', 'Problem-Solving',
    'Collaboration', 'Ethics', 'Creativity', 'Leadership'
]

class IntentParser:
    """
    Maps user text to (command, params, core_area).
    """
    def parse(self, text: str) -> tuple[str, dict, str]:
        # Rule-based or ML-based intent matching
        return 'unknown', {}, CORE_AREAS[0]
```

src/kernel/kernel.py

```
from .config import ConfigManager
from .storage import StorageAdapter

class PlayNACKernel:
    """
    Core orchestrator: ingest → auth → intent → modules → review/advisory.
    """
    def __init__(self, config: ConfigManager, storage: StorageAdapter):
        self.config = config; self.config.load()
        self.storage = storage
        self.ep = EarnedPathEngine()
        self.auth = BiometricAuth()
        self.advisor = ExpertAdvisor([])
        self.review = PeerReviewEngine()
        self.media = MediaProcessor()
        self.intent = IntentParser()

    def handle_message(self, user_id: str, text: str, bio_sample: bytes) -> Any:
        if not self.auth.verify(bio_sample):
```

```
    raise PermissionError("Human verification failed")
cmd, params, area = self.intent.parse(text)
# route command to EP, review, advisory, or media
pass
```

✅ Recommended Changes-Edits Summary

1. **Simplified** EarnedPath: binary skill unlocks → credentials.
2. **Replaced** BioPoW with **BiometricAuth** for proof-of-human.
3. **Collapsed** 17×7 matrix to **7 core areas** for simplicity.
4. **GAIA** → **ExpertAdvisor**: advisory, not governance.
5. **JASConsensus** → **PeerReviewEngine** for quality assurance.
6. **MediaProcessor** refocused on creative feedback.
7. **Removed** token economics, GiantERP, mystical and complex scoring.
8. **Docs** updated: docs/overview.md, docs/core-areas.md.
9. **Examples**: demo scripts reflect human-centric flow.
10. **CHANGELOG** records v7.0→v7.1 transitions.

This streamlined codebase now empowers developers to build a human-centered skill platform: verifiable credentials, expert guidance, community validation, and anti-bot assurance—all with minimal complexity.