ERES Institute for New Age Cybernetics ~ PlayNAC "KERNEL" Codebase V7.3
Empirical Realtime Education System × Human-Centered Skill Development Platform

Key Updates in V7.3:
- Ingestion & Sync:
  • Flesh out src/utils/ingestion/ with real API clients, rate-limit handling, retry logic
  • researchgate.py, medium.py, github_sync.py implemented with OAuth and caching
- Context Manager:
  • Stateful session/user-scoped context in src/kernel/context_manager.py
  • Support for multi-turn Q&A, intent chaining, and EP node linkage
- Vacationomics Engine:
  • New module src/vacationomics/ encapsulating time-budget simulations
  • Implements GCF tradeoff calculations (UBI vs. merit)
- Error Handling & Observability:
  • Expanded utils/exceptions.py with domain-specific errors
  • Structured logging via utils/logger.py with request/session IDs
  • Prometheus-compatible metrics stub in utils/metrics.py
- Tests & CI/CD:
  • New tests/ingestion, tests/vacationomics, high coverage targets
  • GitHub Actions workflow for lint, mypy, pytest, bandit, coverage badges
- Type Safety & Documentation:
  • Full type hints across all modules
  • Sphinx autodoc configured in docs/ for Google-style docstrings
  • Architecture diagrams in docs/architecture/
- Configuration & Deployment:
  • Docker Compose orchestration (kernel + mock GERP + NBERS)
  • Kubernetes Helm chart skeleton in deploy/helm/
- Real-World Integrations:
  • AuraScanner adapters for Muse/OpenBCI in src/bee/
  • Three.js example stub in examples/greenbox/
  • ASR/TTS adapters for Google, Azure, Whisper in src/nav/
- Performance & Scalability:
  • Async I/O stubs for GERP and ingestion modules (asyncio)
  • Batching & caching with aiocache/Redis interface
  • Benchmark scripts in bench/ measuring simulation latency

Directory Structure
~~~
```text
.
├── src/
~~~

```
│   ├── kernel/
│   │   ├── config.py          # ConfigManager supports multiple env files
│   │   ├── context_manager.py # Stateful, session-scoped context store
│   │   └── playnac_kernel.py  # Orchestrator updated for new engines
│   ├── utils/
│   │   ├── exceptions.py      # Domain-specific error classes
│   │   ├── logger.py          # Structured logger with contexts
│   │   ├── metrics.py         # Prometheus metrics stubs
│   │   └── ingestion/
│   │       ├── researchgate.py
│   │       ├── medium.py
│   │       └── github_sync.py
│   ├── vacationomics/
│   │   └── simulation.py      # Time-budget & GCF tradeoffs
│   ├── earnedpath/
│   ├── gianterp/
│   ├── bee/
│   ├── berc/
│   ├── media/
│   ├── nav/
│   └── ...
├── tests/
│   ├── ingestion/
│   ├── vacationomics/
│   └── kernel/
├── docs/
│   ├── architecture/
│   └── api/
├── deploy/
│   ├── docker-compose.yml
│   └── helm/
├── bench/
└── .github/
    └── workflows/ci.yml
```

# src/utils/ingestion/researchgate.py
```python
import time
from typing import List, Dict
from ..exceptions import IngestionError
from ..utils.logger import get_logger
class ResearchGateClient:
    def __init__(self, oauth_token: str, rate_limit: float = 1.0):
        self.logger = get_logger(__name__)
        self.token = oauth_token
```

~~~
        self.rate_limit = rate_limit
    def fetch_publications(self, user_id: str) -> List[Dict]:
        """Fetch and normalize publications from ResearchGate API"""
        try:
            # Real OAuth request with retry
            # time.sleep(self.rate_limit) and handle rate-limit headers
            pass
        except Exception as e:
            self.logger.error(f"RG fetch failed for {user_id}: {e}")
            raise IngestionError("ResearchGate ingestion error")
```

---
# src/kernel/context_manager.py
```python
from typing import Dict, Any
class ContextManager:
    """Maintains per-session state across modules for multi-turn
interactions."""
    def __init__(self):
        self.sessions: Dict[str, Dict[str, Any]] = {}
    def get_context(self, session_id: str) -> Dict[str, Any]:
        return self.sessions.setdefault(session_id, {})
    def update_context(self, session_id: str, key: str, value: Any) ->
None:
        self.get_context(session_id)[key] = value
    def clear_session(self, session_id: str) -> None:
        self.sessions.pop(session_id, None)
```

---
# src/vacationomics/simulation.py
```python
from typing import Dict, Any
class VacationomicsEngine:
    """Time-budget and UBI-merit tradeoff simulations via GCF"""
    def __init__(self, config: Dict[str, Any]):
        self.config = config
    def simulate(self, user_id: str, hours: float) -> Dict[str, Any]:
        """Run a time vs. merit vs. resource budget scenario."""
        # Compute EP reward and resource cost
        pass
    def calculate_tradeoff(self, ep: float, capacity: float) -> float:
        """Graceful Contribution Formula: Meritcoin = α*EP +
β*(capacity/max)"""
        alpha = float(self.config.get('GCF_ALPHA', 10))
        beta = float(self.config.get('GCF_BETA', 0.1))
~~~

~~~
        return alpha * ep + beta * (capacity / 100)
```

---
# src/utils/exceptions.py
```python
class KernelError(Exception):
    """Base exception for PlayNAC Kernel errors"""
class IngestionError(KernelError):
    """Errors during content ingestion/sync"""
class VacEngineError(KernelError):
    """Errors in Vacationomics Engine"""
# Add domain-specific: GERPClientError, BioPoWError, HFVNModeError, etc.
~~~