**Part 1: BACKEND CODE GENERATION PROMPT**

**Project Overview (Backend Focus):**

Develop the backend for a next-generation flight training management system that surpasses current solutions (e.g., Hinfact, SimOrg) by offering intelligent scheduling, document processing, compliance, real-time analytics, adaptive assessments, and advanced AI integrations. The backend is built as a set of microservices using Modern C++ (C++17/20) with the Drogon framework and Python for AI/ML tasks. It must support high performance, low latency (including real-time simulator telemetry at 1000Hz), robust security (AES-256, TLS 1.3, zero-trust), and scalable API endpoints. The overall repository structure for the backend is as follows:

/advanced-pilot-training-platform

/backend

/core # Shared utilities (configuration, logging, error handling)

/document # Document processing pipeline and AI-based content extraction

/syllabus # Syllabus generation engine and training structure creation

/assessment # Competency-based assessment, grading, and biometric integrations

/user-management # Authentication, digital logbooks, and role-based dashboards

/scheduler # AI-driven scheduling and resource optimization module

/analytics # Real-time performance analytics and predictive insights

/compliance # Regulatory compliance engine, audit trails, and document verification

/collaboration # Backend support for virtual workspaces and messaging integration

/visualization # Data services for 3D/AR knowledge maps and simulation visualizers

/integration # Connectors for simulators, biometric devices, enterprise systems, and calendars

/security # Zero-trust security, blockchain audit trails, and ethical AI governance

**Instructions:**

1. **Core Framework Components:**
   * Generate a ConfigurationManager class to load settings from environment, files, and database with type-safe access and change notifications.
   * Include robust logging (structured and contextual) and thread-safe error handling using modern C++ practices.
2. **Document Processing Pipeline:**
   * Create an abstract DocumentProcessor interface with concrete implementations for handling PDF, DOCX, XLSX, HTML, and PPTX.
   * Integrate OCR (e.g., via Tesseract), ML-based structure recognition, and entity extraction.
   * Implement asynchronous processing with progress tracking and error reporting.
   * Ensure the pipeline supports regulatory mapping and audit logging.
3. **Syllabus Generation Engine:**
   * Develop a SyllabusGenerator class that processes document analyses to extract learning objectives, competency areas, and training requirements.
   * Support template-based syllabus creation with customization, version control, and audit trails.
   * Map regulatory standards (FAA, EASA, ICAO) to syllabus components.
4. **Real-Time Data Processing:**
   * Create a SimulatorDataProcessor that handles high-frequency telemetry (1000Hz) using lock-free queues and multithreading.
   * Integrate SIMD optimizations for processing simulator data.
   * Provide both real-time and historical data access along with anomaly detection.
5. **API Gateway and RESTful Endpoints:**
   * Build a comprehensive API gateway using the Drogon framework with:
     + JWT-based authentication, input validation, and rate limiting.
     + Modular controllers for each backend module (document, syllabus, assessment, scheduler, etc.).
     + Auto-generated OpenAPI/Swagger documentation.
   * Ensure detailed error handling, logging, and monitoring.
6. **Database Access Layer:**
   * Implement a DatabaseManager with connection pooling for PostgreSQL (and TimescaleDB for time-series data).
   * Include prepared statement caching, transaction management, and migration support.
   * Support structured queries and time-series data operations.
7. **AI & ML Modules (Python):**
   * Develop pipelines for document understanding:
     + Document classification, text summarization, named entity recognition, and relationship extraction using transformer models.
   * Create performance prediction models (using TensorFlow/PyTorch) to forecast trainee outcomes and suggest adaptive interventions.
   * Build no-code automation workflows (e.g., auto-generation of tasks from document insights) and auto-research assistant modules with web scraping, citation tracking, and plagiarism checks.
8. **Security & Compliance:**
   * Integrate zero-trust security measures: AES-256 encryption, TLS 1.3 for data in transit, multi-factor authentication (MFA), and blockchain-backed audit trails.
   * Implement ethical AI governance with bias detection, transparency reporting, and user-controlled data ownership.
   * Ensure comprehensive audit logging for regulatory compliance.
9. **Testing and Performance:**
   * Include unit tests (using Google Test) for each module.
   * Write integration tests covering end-to-end workflows (document ingestion through syllabus generation).
   * Benchmark real-time components and optimize for latency (target <5ms response for high-frequency data APIs).
10. **Documentation and CI/CD:**
    * Provide inline documentation and developer guides for each module.
    * Generate API documentation using OpenAPI standards.
    * Set up CI/CD pipelines (via GitHub Actions) for automated testing, linting, and deployment to Vercel (if containerized microservices are deployed there).

**Final Note:**  
The backend code generated should be fully modular, thoroughly tested, and follow best practices in error handling, performance optimization, and security. Once complete, this code will integrate seamlessly with the frontend services.