**Objective**

Generate the React TypeScript components that form the user interface of the Advanced Pilot Training Platform, providing intuitive tools for document management, syllabus building, assessment, and analytics.

**Requirements**

**1. Component Library Foundation**

Create a set of base UI components including:

* Button, Input, Select with proper accessibility
* Form components with validation integration
* Data visualization components (charts, graphs)
* Navigation components (tabs, pagination)
* Feedback components (alerts, notifications)
* Modal and dialog components
* Responsive layout components (grid, flex containers)
* Data table with sorting, filtering, and pagination
* Calendar and scheduling components
* Rich text editor with formatting options
* File upload with preview capabilities
* Progress indicators and loaders
* Theme system with light and dark mode support

**2. Syllabus Builder Interface**

Create a SyllabusBuilder component that:

* Supports drag-and-drop of modules and lessons
* Provides an intuitive tree-view of the syllabus structure
* Allows inline editing of syllabus elements
* Shows regulatory compliance status visually
* Enables template application and customization
* Provides visual indicators for element relationships
* Supports bulk operations on multiple elements
* Includes timeline view of training program
* Offers split-view for comparing versions
* Provides context-sensitive help and suggestions
* Includes revision history with diff visualization
* Supports collaborative editing with presence indicators
* Implements customizable views (compact, detailed, print)

**3. Document Upload and Management**

Create a DocumentManagement component that:

* Supports drag-and-drop file uploads with progress tracking
* Handles batch uploads with status tracking
* Provides document preview capabilities
* Shows document processing status and results
* Enables document organization and categorization
* Includes versioning and comparison tools
* Provides full-text search with highlighted results
* Supports document annotation and commenting
* Includes document workflows with approval stages
* Offers bulk operations (tag, move, delete, export)
* Implements access control visualization
* Provides usage analytics and insights
* Supports offline access to critical documents

**4. Assessment and Grading Interface**

Create an AssessmentInterface component that:

* Implements one-click grading on a 1-4 scale
* Provides competency-based assessment forms
* Shows performance trends and comparisons
* Enables digital signature capture
* Supports offline assessment with synchronization
* Includes customizable assessment templates
* Offers speech-to-text for assessment comments
* Provides performance visualization against standards
* Supports photo/video capture for evidence
* Includes student self-assessment capabilities
* Offers side-by-side comparison with previous assessments
* Generates PDF reports for official documentation
* Implements instructor-student feedback loop

**5. Analytics Dashboard**

Create an AnalyticsDashboard component that:

* Displays key performance indicators with drill-down
* Shows training program effectiveness metrics
* Provides compliance status visualization
* Enables custom report generation
* Supports both fleet-wide and individual trainee views
* Includes predictive analytics visualizations
* Offers customizable dashboard layouts
* Provides real-time updating of critical metrics
* Supports comparison across time periods
* Includes export capabilities for reports
* Implements alerting for significant deviations
* Offers cohort analysis visualization
* Provides resource utilization insights
* Supports interactive data exploration

**Implementation Guidelines**

* Use TypeScript with strict type checking
* Implement responsive design for all components
* Follow accessibility guidelines (WCAG 2.1 AA)
* Use React hooks for state management
* Implement efficient rendering with memoization
* Design components with reusability in mind
* Include unit tests for all components using Jest and React Testing Library
* Use Storybook for component documentation and development
* Implement proper error boundaries for component failure isolation
* Design for performance with virtualization for large datasets
* Follow atomic design principles for component organization
* Implement keyboard shortcuts for power users
* Support internationalization and localization
* Use proper code splitting for optimized loading
* Include analytics for UI usage patterns