 I've created the core subscription-related components, let me summarize what I've built:

1. **Models**:
   * **subscription.model.js**: Defines the subscription schema with plan types (Community, Premium, Enterprise), features, limits, and billing information
   * **subscription-history.model.js**: Tracks all changes to subscriptions for audit purposes
2. **Services**:
   * **subscription.service.js**: Handles all subscription-related business logic including:
     + Creating and updating subscriptions
     + Processing renewals
     + Feature access validation
     + Integration with payment and notification services
3. **Controllers & Routes**:
   * **subscription.controller.js**: Handles HTTP requests for subscription operations
   * **subscription.routes.js**: Defines API endpoints with proper authentication and authorization
4. **Integrations**:
   * **payment.service.js**: Handles payment processing for subscriptions
   * **notification.service.js**: Manages notifications for subscription events
5. **Validation**:
   * **subscription.schema.js**: Defines validation schemas for subscription-related requests

The system supports all the subscription tiers you specified:

1. **Community Edition**:
   * Limited to 500 students
   * 50GB storage
   * Basic features
2. **Premium Edition**:
   * Up to 2,000 students
   * 500GB storage
   * Advanced features including AI analytics and API access
3. **Enterprise Edition**:
   * Unlimited users and storage
   * Full feature access including white-labeling and custom development

Key Features:

* Full subscription lifecycle management (create, update, cancel, renew)
* Automatic renewal processing
* Feature access control based on subscription tier
* Payment integration
* Notification system for subscription events
* Detailed subscription history tracking
* Tenant-aware design