1. **Repository Pattern Implementation**
   * Create repository interfaces in the Domain layer for each entity
   * Implement these repositories in the Infrastructure layer
   * This will provide a clean abstraction for data access
2. **Service Layer Implementation**
   * Create service interfaces in the Application layer
   * Implement business logic in services
   * This layer will handle the application's core functionality
3. **API Controllers**
   * Create REST API controllers for each entity
   * Implement CRUD operations
   * Add proper routing and action methods
   * Implement proper error handling and responses
4. **Authentication and Authorization**
   * Implement user authentication (likely using JWT tokens)
   * Set up role-based authorization
   * Configure security policies
5. **DTOs (Data Transfer Objects)**
   * Create DTOs for each entity to separate domain models from API responses
   * Implement AutoMapper for object mapping
   * This will help in data transformation and API response shaping
6. **Validation**
   * Implement request validation using FluentValidation
   * Add custom validation rules
   * Handle validation responses
7. **Middleware**
   * Implement exception handling middleware
   * Add request/response logging
   * Configure CORS policies
8. **Unit Tests**
   * Write unit tests for services
   * Test repository implementations
   * Add integration tests for API endpoints