Podcast App Documentation

C# ASP.NET Core MVC Project presented by Carl Nicolas Mendoza and Neil Hontanos for Lab 3.

Project Structure

Models/

Contains data entities and view models.

Entity Models (map to database tables):

- `Podcast.cs` - Podcast entity
- `Episode.cs` - Episode entity
- `User.cs` - User entity (extends IdentityUser)
- `Subscription.cs` - Subscription entity
- `Comment.cs` - Comment entity (for DynamoDB mapping)

View Models:

- `RegisterViewModel.cs` - User registration form
- `LoginViewModel.cs` - Login form
- `PodcastViewModel.cs` - Podcast creation/editing
- `EpisodeViewModel.cs` - Episode creation/editing
- `EpisodeDetailsViewModel.cs` - Episode display with comments
- `CommentViewModel.cs` - Comment display/creation
- `AnalyticsViewModel.cs` - Dashboard statistics

Enums:

- `UserRole.cs` - Enum for Podcaster, Listener, Admin

Data/

Database contexts and repository pattern implementation.

Database Contexts:

- `ApplicationDbContext.cs` - EF Core context for SQL Server (Podcasts, Episodes, Users, Subscriptions)

Repositories (Interface + Implementation):

- `IPodcastRepository.cs` / `PodcastRepository.cs`
- `IEpisodeRepository.cs` / `EpisodeRepository.cs`
- `ISubscriptionRepository.cs` / `SubscriptionRepository.cs`
- `ICommentRepository.cs` / `CommentRepository.cs` - For DynamoDB operations
- `IUserRepository.cs` / `UserRepository.cs`

Controllers/

Handle HTTP requests and coordinate between services and views.

Controllers:

- `AccountController.cs` - Authentication (register, login, logout)
- `PodcastController.cs` - Podcast CRUD operations

- `EpisodeController.cs` - Episode CRUD, viewing, search
- `CommentController.cs` - Comment CRUD operations
- `SubscriptionController.cs` - Subscribe/unsubscribe functionality
- `AnalyticsController.cs` - Dashboard and reporting
- `AdminController.cs` - User management, episode approval
- `HomeController.cs` - Landing page, about, contact

Views/

Razor view templates for rendering HTML.

Folder Structure:

```
Views/
├── Shared/
│   ├── _Layout.cshtml - Master layout
│   ├── _LoginPartial.cshtml - Login/logout links
│   └── _ValidationScriptsPartial.cshtml
├── Home/
│   ├── Index.cshtml - Landing page
│   └── About.cshtml
├── Account/
│   ├── Register.cshtml
│   ├── Login.cshtml
│   └── AccessDenied.cshtml
├── Podcast/
│   ├── Index.cshtml - List all podcasts
│   ├── Create.cshtml
│   ├── Edit.cshtml
│   ├── Delete.cshtml
│   └── Details.cshtml
├── Episode/
│   ├── Index.cshtml - Browse/search episodes
│   ├── Create.cshtml
│   ├── Edit.cshtml
│   ├── Delete.cshtml
│   └── Details.cshtml - Episode player + comments
├── Analytics/
│   └── Dashboard.cshtml
└── Admin/
    ├── Index.cshtml
    ├── ManageUsers.cshtml
    └── ApproveEpisodes.cshtml
```

Services/

Business logic layer (optional but recommended for clean architecture).

Service Interfaces + Implementations:

- `IS3Service.cs` / `S3Service.cs` - S3 file upload/download
- `IDynamoDBService.cs` / `DynamoDBService.cs` - DynamoDB operations
- `IEpisodeService.cs` / `EpisodeService.cs` - Episode business logic
- `IPodcastService.cs` / `PodcastService.cs` - Podcast business logic
- `IAnalyticsService.cs` / `AnalyticsService.cs` - Analytics aggregation

wwwroot/

Static files served directly to the client.

```
wwwroot/
├── css/
│   └── site.css - Custom styles
├── js/
│   └── site.js - Custom JavaScript
├── lib/ - Third-party libraries (Bootstrap, jQuery)
└── images/
    └── (podcast thumbnails, logos, etc.)
```

Properties/

Configuration files.

- `launchSettings.json` - Development server settings

Root Files

- `Program.cs` - Application entry point, service configuration
- `appsettings.json` - Configuration (connection strings, AWS settings)
- `appsettings.Development.json` - Development-specific settings
- `.gitignore` - Exclude bin/, obj/, appsettings.Development.json
- `PodcastApp.csproj` - Project file with NuGet packages

Lab Requirements Summary

System Architecture

The application follows a layered architecture:

1. **Presentation Layer**: ASP.NET MVC views for user interfaces
2. **Business Logic Layer**: Controllers and services
3. **Data Access Layer**: Repositories for database interactions

Database Design

Relational Database (SQL Server)

- **Podcasts Table:** PodcastID, Title, Description, CreatorID, CreatedDate
- **Episodes Table:** EpisodeID, PodcastID, Title, ReleaseDate, Duration, PlayCount, AudioFileURL, Views
- **Users Table:** UserID, Username, Email, Role (ASP.NET Identity)
- **Subscriptions Table:** SubscriptionID, UserID, PodcastID, SubscribedDate

DynamoDB (Unstructured Data)

- **Comments Table:** EpisodeID, PodcastID, CommentID, UserID, Text, Timestamp

Features to Implement

1. **User Authentication**
   - Register/login as Podcaster/Listener/Admin
   - Role-based access control
2. **Podcast Management** (Podcaster Role)
   - Create/edit/delete podcasts and episodes
   - Upload audio/video files to S3
   - Metadata stored in SQL Server
3. **Episode Viewing/Interaction** (Listener Role)
   - Browse/search episodes (SQL queries)
   - Add/edit comments (DynamoDB)
   - Subscribe to podcasts
4. **Analytics Dashboard** (Admin/Podcaster)
   - View episode stats
   - Top episodes by views
   - Aggregate data from SQL + DynamoDB
5. **Admin Panel**
   - Manage users
   - Approve episodes

Required AWS Services

- **AWS Elastic Beanstalk -** Deploy and host the ASP.NET Core MVC app
- **Amazon DynamoDB -** Store unstructured data (comments)
- **Amazon S3 -** Store audio/video files
- **AWS Systems Manager Parameter Store -** Securely store RDS credentials
- **AWS IAM -** Manage resource access and roles

NuGet Packages Required

xml

```xml
<!-- Entity Framework Core for SQL Server -->
<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.0" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.0" />

<!-- ASP.NET Core Identity -->
<PackageReference Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore" Version="8.0.0" />

<!-- AWS SDK -->
<PackageReference Include="AWSSDK.S3" Version="3.7.0" />
<PackageReference Include="AWSSDK.DynamoDBv2" Version="3.7.0" />
<PackageReference Include="AWSSDK.Extensions.NETCore.Setup" Version="3.7.0" />
<PackageReference Include="AWSSDK.SimpleSystemsManagement" Version="3.7.0" />
```

Getting Started

Step 1: Set Up Models

Define your entities in `Models/` folder based on the database schema.

Step 2: Configure Database Context

Create `ApplicationDbContext.cs` in `Data/` folder with DbSet properties for each entity.

Step 3: Implement Repositories

Create repository interfaces and implementations for data access patterns.

Step 4: Build Controllers

Implement CRUD operations and business logic in controllers.

Step 5: Create Views

Build Razor views for each controller action.

Step 6: AWS Integration

- Configure S3 for file uploads
- Set up DynamoDB tables
- Configure IAM roles
- Store credentials in Parameter Store

Step 7: Deploy to Elastic Beanstalk

Package and deploy the application to AWS.

Development Workflow

1. **Local Development:**
   - Use LocalDB or SQL Server Express
   - Use DynamoDB Local for testing
   - Use LocalStack for S3 simulation (optional)
2. **Configuration:**
   - Store sensitive data in `appsettings.Development.json` (excluded from git)
   - Use AWS Parameter Store in production
3. **Testing:**
   - Unit tests for services
   - Integration tests for repositories
   - Manual testing of UI flows

Security Considerations

- Use ASP.NET Core Identity for authentication
- Implement authorization with [Authorize] attributes
- Validate all user inputs
- Use HTTPS in production
- Secure AWS credentials with IAM roles
- Implement CSRF protection (built-in with MVC)
- Sanitize user-generated content (comments)