NCDB Development Roadmap & Master Checklist

Nicolas Cage Database - iOS 26 App with Liquid Glass Design

📋 PHASE 0: PRE-DEVELOPMENT SETUP

Step 1: Export Design Documentation & Code Excerpts

Purpose: Create reference files from project discussions for Claude Code CLI

Tasks:

- ☐ Export all SwiftData model definitions to reference files
- ☐ Export TMDb integration architecture & code
- ☐ Export Liquid Glass UI component examples
- ☐ Export onboarding flow specifications
- ☐ Export widget designs & configurations
- ☐ Export news scraper implementation details
- ☐ Export achievements/gamification system specs
- ☐ Export social sharing templates & logic
- ☐ Export settings screen structure
- ☐ Export website export/FTP integration code

Deliverables:

```
/Documentation/
  ├─ Data_Models.md
  ├─ TMDb_Integration.swift
  ├─ LiquidGlass_Components.swift
  ├─ Onboarding_Flow.md
  ├─ Widget_Specifications.md
  ├─ News_Scraper.swift
  ├─ Achievements_System.md
  ├─ Social_Sharing.swift
  ├─ Settings_Structure.md
  └─ Export_System.swift
```

Step 2: Xcode Project Setup

Purpose: Initialize project with proper configuration for iOS 26 & SwiftUI

2.1 Create New Xcode Project

Configuration:

- **Platform:** iOS

- **Template:** App
- **Interface:** SwiftUI
- **Language:** Swift
- **Minimum Deployment:** iOS 26.0
- **Organization Name:** [Your Organization]
- **Bundle Identifier:** com.[yourorg].ncdb

2.2 Project Structure Setup

Create folder structure:

```
NCDB/
├── App/
│   ├── NCDBApp.swift (main app entry)
│   └── AppDelegate.swift (if needed)
├── Models/
│   ├── Production.swift
│   ├── CastMember.swift
│   ├── WatchEvent.swift
│   ├── ExternalRating.swift
│   ├── CustomTag.swift
│   ├── NewsArticle.swift
│   ├── Achievement.swift
│   ├── ExportTemplate.swift
│   └── UserPreferences.swift
├── ViewModels/
│   ├── HomeViewModel.swift
│   ├── MovieListViewModel.swift
│   ├── MovieDetailViewModel.swift
│   ├── RankingViewModel.swift
│   └── StatsViewModel.swift
├── Views/
│   ├── Home/
│   ├── MovieList/
│   ├── MovieDetail/
│   ├── Rankings/
│   ├── Stats/
│   ├── Settings/
│   └── Onboarding/
├── Services/
│   ├── TMDbService.swift
│   ├── CacheManager.swift
│   ├── NewsScraper.swift
│   ├── ExportService.swift
│   └── AchievementManager.swift
```

```
├── Components/
│   ├── LiquidGlass/
│   │   ├── GlassCard.swift
│   │   ├── GlassButton.swift
│   │   ├── GlassFrame.swift
│   │   └── GoldBadge.swift
│   └── Custom/
│       ├── StarRating.swift
│       ├── MoviePosterCard.swift
│       └── RankingCarousel.swift
├── Utilities/
│   ├── Constants.swift
│   ├── Extensions/
│   └── Helpers/
├── Resources/
│   ├── Assets.xcassets/
│   ├── PreloadedData/
│   │   ├── face_off.json
│   │   └── con_air.json
│   └── Fonts/ (if custom fonts)
└── Widgets/
    └── NCDBWidgets/
```

2.3 Add Dependencies (Swift Package Manager)

Required Packages:

- [ ] Add FeedKit (for RSS news scraping)
  - Repository: `https://github.com/nmdias/FeedKit`
- [ ] Add any networking helpers if needed

Optional Packages:

- [ ] Kingfisher (image caching) - consider if needed beyond custom solution
- [ ] SwiftUI Introspect (if needed for advanced customization)

2.4 Configure Build Settings

- [ ] Set iOS deployment target to 26.0
- [ ] Enable SwiftUI previews
- [ ] Configure code signing
- [ ] Set up app capabilities:
- [ ] Background modes (if needed for news updates)
- [ ] Push notifications (optional)
- [ ] App Groups (for widget data sharing)

2.5 Working with Claude Code + Xcode AI

Hybrid Development Approach:

Hybrid Development Approach:

Claude Code CLI Usage:

- Use for complex logic implementation
- Architecture-level decisions
- Batch file generation
- Code review and refactoring

Xcode AI Coding Assistant Usage:

- Use for inline code completion
- Quick fixes and suggestions
- UI layout tweaking in real-time
- SwiftUI preview debugging

Recommended Workflow:

1. **Plan** with Claude Code: Discuss architecture, generate service layer code
2. **Implement** in Xcode: Copy code, use Xcode AI for refinements
3. **Test** in Xcode: Build, run, iterate with Xcode AI
4. **Review** with Claude Code: Get feedback on implementation
5. **Refine** in Xcode: Apply suggestions, continue with AI assistance

Tips for Hybrid Mode:

- Keep Claude Code terminal open in separate window
- Use Claude Code for generating complete files
- Use Xcode AI for completing partial implementations
- Commit frequently to Git so both tools can track changes
- Use Claude Code for explaining complex existing code

🏗️ PHASE 1: CORE FOUNDATION (Week 1-2)

Milestone 1.1: Data Layer Setup

Goal: Establish SwiftData models and persistence

- ▢ Create all SwiftData model files
- ▢ Production model with all properties
- ▢ CastMember model with relationships
- ▢ WatchEvent model for tracking views

- [ ] ExternalRating model
- [ ] CustomTag model
- [ ] NewsArticle model
- [ ] Achievement model
- [ ] ExportTemplate model
- [ ] UserPreferences model
- [ ] Set up ModelContainer in App entry point
- [ ] Create sample data for testing
- [ ] Implement model relationships and cascading deletes
- [ ] Test data persistence with SwiftData

Test Checkpoint: Create and persist a movie, verify it appears after app restart

---

Milestone 1.2: TMDb Service Integration

Goal: Connect to TMDb API and fetch movie data

- [ ] Create TMDbService class
- [ ] Implement API key storage in Keychain
- [ ] Build basic networking layer
- [ ] Error handling
- [ ] Rate limiting (4 req/sec)
- [ ] Response parsing
- [ ] Implement core endpoints:
- [ ] Fetch Nicolas Cage filmography
- [ ] Fetch movie details
- [ ] Fetch movie posters/images
- [ ] Search movies
- [ ] Create CacheManager for offline support
- [ ] ETag/Last-Modified support
- [ ] Cache expiry logic
- [ ] Three-tier cache (Essential/Moderate/Ephemeral)
- [ ] Bundle 2 preloaded movies (Face/Off, Con Air)
- [ ] Test API calls and caching

Test Checkpoint: Fetch Nic Cage movies, view offline, verify cache works

---

Milestone 1.3: Basic UI Foundation

Goal: Implement Liquid Glass design system components

- [ ] Create color scheme constants (Cage Gold #FFD700, blacks, whites)
- [ ] Build reusable Liquid Glass components:
- [ ] GlassCard view

- [ ] GlassButton view
- [ ] GlassFrame for posters
- [ ] GoldBadge view
- [ ] Implement frosted material backgrounds
- [ ] Create custom modifiers for glass effects
- [ ] Test components in SwiftUI previews

Test Checkpoint: Display glass cards with different content, verify visual consistency

---

🎬 PHASE 2: CORE FEATURES (Week 3-4)

Milestone 2.1: Movie List & Detail Views

Goal: Display movies and detailed information

- [ ] Implement MovieListView
- [ ] Display all Nicolas Cage movies
- [ ] Filter by watched/unwatched
- [ ] Search functionality
- [ ] Sort options (year, rating, title)
- [ ] Liquid Glass styling
- [ ] Implement MovieDetailView
- [ ] Movie poster with glass frame
- [ ] All movie metadata
- [ ] Cast members display
- [ ] External ratings (IMDb, RT)
- [ ] Watch status toggle
- [ ] Star rating input
- [ ] Review text field
- [ ] Add to custom tags
- [ ] Create MovieListViewModel
- [ ] Create MovieDetailViewModel
- [ ] Implement navigation between views

Test Checkpoint: Browse movies, tap to see details, mark as watched, add rating

---

Milestone 2.2: Watch Tracking & Ratings

Goal: Track viewing history and ratings

- [ ] Implement watch status tracking
- [ ] Mark as watched/unwatched
- [ ] Record watch date automatically
- [ ] Support rewatch tracking

- [ ] Build star rating component (5-star system)
- [ ] Implement rating storage and retrieval
- [ ] Create rating statistics calculations
- [ ] Display rating distribution in stats

Test Checkpoint: Rate 5 movies, verify statistics update correctly

---

Milestone 2.3: Home/Dashboard View

Goal: Create engaging home screen with stats and news

- [ ] Implement HomeView layout
- [ ] Welcome message
- [ ] Quick stats cards (glass panels)
  - Total watched count
  - Average rating
  - Recently watched
  - Next to watch suggestions
- [ ] News feed integration
- [ ] Featured movie of the day
- [ ] Quick actions (add review, view rankings)
- [ ] Create HomeViewModel
- [ ] Implement news scraper service
- [ ] RSS feed parsing (FeedKit)
- [ ] Filter Nicolas Cage articles
- [ ] Store in NewsArticle model
- [ ] Update frequency (daily)
- [ ] Display news articles in feed

Test Checkpoint: View home screen with stats and news, navigate to features

---

🏆 PHASE 3: ADVANCED FEATURES (Week 5-6)

Milestone 3.1: Interactive Rankings System

Goal: Build unique ranking carousel with drag-and-drop

- [ ] Create RankingCarousel component
- [ ] Horizontal scrolling card layout
- [ ] Drag-and-drop reordering
- [ ] Smooth animations
- [ ] Position labels (#1, #2, etc.)
- [ ] Visual depth with glass effects

- [ ] Implement ranking persistence

- [ ] Create RankingViewModel
- [ ] Build RankingsView
- [ ] Display current rankings
- [ ] "Start Ranking" button
- [ ] Filter ranked vs unranked
- [ ] Export rankings option
- [ ] Add ranking change animations
- [ ] Implement tie-breaking logic

Test Checkpoint: Rank 10 movies, reorder them, verify persistence

---

Milestone 3.2: Statistics Dashboard

Goal: Comprehensive stats and analytics

- [ ] Create StatsView
- [ ] Overview card (total watched, rated, reviewed)
- [ ] Rating distribution chart
- [ ] Watch frequency graph
- [ ] Genre breakdown chart
- [ ] Decade analysis
- [ ] Favorite actors/directors
- [ ] Longest/shortest movies watched
- [ ] Rewatch statistics
- [ ] Create StatsViewModel with calculations
- [ ] Implement chart components (use Charts framework)
- [ ] Add filter options (by year, genre, rating)
- [ ] Style with Liquid Glass aesthetic

Test Checkpoint: View stats with sample data, verify accuracy of calculations

---

Milestone 3.3: Custom Tags & Collections

Goal: Organize movies with custom collections

- [ ] Implement CustomTag model functionality
- [ ] Create TagsView
- [ ] Display all custom tags
- [ ] Add new tag UI
- [ ] Edit/delete tags
- [ ] View movies in each tag
- [ ] Add tag management to MovieDetailView
- [ ] Implement tag-based filtering

- [ ] Create pre-defined tags (Action, Drama, Comedy, etc.)
- [ ] Support multiple tags per movie

Test Checkpoint: Create tags, assign movies, filter by tag

🎯 PHASE 4: USER EXPERIENCE (Week 7-8)

Milestone 4.1: Onboarding Flow

Goal: Smooth first-launch experience

- [ ] Create OnboardingCoordinator
- [ ] Build onboarding screens:
- [ ] Splash screen
- [ ] Welcome screen
- [ ] Feature highlights (3 swipeable screens)
- [ ] TMDb API key setup screen
- [ ] Initial data loading screen with progress
- [ ] Actor selection screen (Nic Cage pre-selected)
- [ ] Ranking tutorial (interactive demo)
- [ ] Permissions request (notifications)
- [ ] "Ready to Go" completion screen
- [ ] Implement skip/next navigation
- [ ] Add progress indicators
- [ ] Store onboarding completion in AppStorage
- [ ] Test complete flow

Test Checkpoint: Complete onboarding, verify API key saved, data loaded

---

Milestone 4.2: Settings & Preferences

Goal: Comprehensive settings management

- [ ] Create SettingsView
- [ ] Account section (TMDb API key management)
- [ ] Appearance section (theme options)
- [ ] Data & Sync section
- [ ] Display preferences
- [ ] Notifications settings
- [ ] Export & backup options
- [ ] About section
- [ ] Advanced/danger zone
- [ ] Implement UserPreferences model
- [ ] Create preference storage logic

- [ ] Build settings screens for each section
- [ ] Add data export/import functionality
- [ ] Implement cache management controls

Test Checkpoint: Change settings, verify they persist and affect app behavior

---

Milestone 4.3: Search & Filters

Goal: Advanced search and filtering capabilities

- [ ] Implement search functionality
- [ ] Search by title
- [ ] Search by year
- [ ] Search by director
- [ ] Search by cast member
- [ ] Create filter UI
- [ ] Filter by watch status
- [ ] Filter by rating range
- [ ] Filter by decade
- [ ] Filter by genre
- [ ] Filter by custom tags
- [ ] Combine search + filters
- [ ] Add sort options
- [ ] Title (A-Z, Z-A)
- [ ] Release year (newest, oldest)
- [ ] Rating (highest, lowest)
- [ ] Recently watched
- [ ] Save filter presets

Test Checkpoint: Search movies, apply filters, verify results are accurate

---

🎮 PHASE 5: GAMIFICATION & SOCIAL (Week 9-10)

Milestone 5.1: Achievements System

Goal: Implement complete achievement tracking

- [ ] Create AchievementManager service
- [ ] Implement all 34 achievements:
- [ ] Watch milestones (First Flight → Cage Completionist)
- [ ] Rating activity achievements
- [ ] Ranking achievements
- [ ] Binge watching achievements
- [ ] Variety achievements

- [ ] Special category achievements
- [ ] Rewatch achievements

- [ ] News engagement achievements
- [ ] Social sharing achievements
- [ ] Completionist achievements
- [ ] Build achievement unlock logic
- [ ] Create achievement notification system
- [ ] Design AchievementCardView with Liquid Glass
- [ ] Build AchievementsView to display all
- [ ] Show progress toward locked achievements
- [ ] Add celebration animations for unlocks

Test Checkpoint: Unlock achievements through actions, verify notifications display

---

Milestone 5.2: Social Sharing Features

Goal: Share reviews, rankings, and stats

- [ ] Create SocialSharingService
- [ ] Implement share templates:
- [ ] Individual review card (Instagram Story format)
- [ ] Top 10 rankings card (Facebook format)
- [ ] Stats milestone card (Twitter format)
- [ ] Year in review card
- [ ] Build image generation logic (9:16, 1:1, 16:9 ratios)
- [ ] Integrate UIActivityViewController (share sheet)
- [ ] Add sharing options to:
- [ ] Movie detail view
- [ ] Rankings view
- [ ] Stats view
- [ ] Implement platform-specific optimizations
- [ ] Add "Copy to Clipboard" option
- [ ] Apply Liquid Glass styling to generated images

Test Checkpoint: Share a review to Photos, verify image renders correctly

---

Milestone 5.3: Notifications

Goal: Optional push notifications for engagement

- [ ] Request notification permissions
- [ ] Implement local notifications:
- [ ] Achievement unlocks

- [ ] New Nicolas Cage news articles
- [ ] "Don't forget to watch" reminders
- [ ] Weekly stats summary

- [ ] Create notification settings in Settings
- [ ] Add notification actions (deep links)
- [ ] Test notification delivery

Test Checkpoint: Receive notification, tap to open relevant screen

---

📱 PHASE 6: WIDGETS & EXTENSIONS (Week 11)

Milestone 6.1: iOS Widgets

Goal: Home screen and Lock Screen widgets

- [ ] Create WidgetKit extension
- [ ] Implement widget configurations:
- [ ] **Small Widget** - Watch progress ring
- [ ] **Medium Widget** - Progress + recent movie
- [ ] **Large Widget** - Dashboard with stats
- [ ] **Lock Screen Circular** - Total watched count
- [ ] **Lock Screen Rectangular** - Last watched movie
- [ ] **Lock Screen Inline** - "X/Y movies watched"
- [ ] Create TimelineProvider for updates
- [ ] Implement widget deep links to app
- [ ] Style with Liquid Glass aesthetic (adapted for widgets)
- [ ] Set up App Groups for data sharing
- [ ] Test widget updates and refresh

Test Checkpoint: Add widgets to home screen, verify data updates correctly

---

🌐 PHASE 7: EXPORT & WEB INTEGRATION (Week 12)

Milestone 7.1: Static Site Export

Goal: Generate HTML website from app data

- [ ] Create ExportService
- [ ] Build HTML templates:
- [ ] Homepage template
- [ ] Movie detail page template
- [ ] Rankings page template
- [ ] Stats page template
- [ ] Create CSS file with Liquid Glass web styling

- [ ] Implement data-to-HTML conversion
- [ ] Add export UI in Settings
- [ ] Implement FTP upload functionality
- [ ] Add Strapi/Netlify integration (optional)
- [ ] Test complete export workflow

Test Checkpoint: Export site, verify HTML files are generated and styled correctly

---

✨ PHASE 8: POLISH & OPTIMIZATION (Week 13-14)

Milestone 8.1: Animations & Transitions

Goal: Smooth, delightful interactions

- [ ] Add view transition animations
- [ ] Implement custom navigation transitions
- [ ] Add micro-interactions:
- [ ] Button press animations
- [ ] Card hover effects (3D Touch if available)
- [ ] Pull-to-refresh animations
- [ ] Loading state animations
- [ ] Refine Liquid Glass visual effects
- [ ] Add haptic feedback for key actions
- [ ] Polish carousel animations
- [ ] Test animation performance

Test Checkpoint: Navigate through app, verify smooth animations throughout

---

Milestone 8.2: Performance Optimization

Goal: Fast, responsive app experience

- [ ] Profile app with Instruments
- [ ] Optimize image loading and caching
- [ ] Reduce memory footprint
- [ ] Optimize database queries
- [ ] Implement lazy loading for large lists
- [ ] Reduce network calls (batch requests)
- [ ] Test on older devices (A13 Bionic minimum)
- [ ] Fix any memory leaks
- [ ] Optimize widget performance

Test Checkpoint: App launches < 2 seconds, smooth scrolling on iPhone 11

Milestone 8.3: Accessibility

Goal: Inclusive experience for all users

- [ ] Add VoiceOver labels to all interactive elements
- [ ] Test complete app with VoiceOver enabled
- [ ] Ensure Dynamic Type support throughout

- [ ] Add accessibility hints where needed
- [ ] Implement sufficient color contrast ratios
- [ ] Add reduce motion alternatives
- [ ] Test with Accessibility Inspector
- [ ] Support keyboard navigation (iPad)

Test Checkpoint: Navigate entire app with VoiceOver, all actions are accessible

---

Milestone 8.4: Testing & Bug Fixes

Goal: Stable, bug-free experience

- [ ] Write unit tests for:
- [ ] TMDb service
- [ ] Data models
- [ ] Cache manager
- [ ] Achievement logic
- [ ] Stats calculations
- [ ] Write UI tests for critical flows:
- [ ] Onboarding
- [ ] Movie rating
- [ ] Ranking movies
- [ ] Exporting data
- [ ] Perform manual testing on all screens
- [ ] Test edge cases (no network, empty states, etc.)
- [ ] Fix all identified bugs
- [ ] Test on multiple device sizes
- [ ] Verify iPad layout (if supporting)

Test Checkpoint: All tests pass, no crashes in typical use

🚀 PHASE 9: PRE-LAUNCH (Week 15)

Milestone 9.1: App Store Preparation

Goal: Ready for TestFlight and App Store submission

- [ ] Finalize app icon (all sizes)
- [ ] Create launch screen
- [ ] Prepare App Store screenshots (6.7", 6.5", 5.5")

- [ ] Write App Store description
- [ ] Create App Store preview video (optional)
- [ ] Set up App Store Connect listing
- [ ] Configure pricing (free with optional tip jar?)
- [ ] Add privacy policy
- [ ] Prepare what's new for version 1.0
- [ ] Test archive and export for distribution

Milestone 9.2: Beta Testing

Goal: Gather feedback and fix issues

- [ ] Distribute TestFlight build
- [ ] Recruit 5-10 beta testers
- [ ] Gather feedback via TestFlight or form
- [ ] Address critical bugs
- [ ] Implement high-priority feedback
- [ ] Test updated build
- [ ] Repeat until stable

Milestone 9.3: Final Release

Goal: Ship version 1.0 to the App Store

- [ ] Create production build
- [ ] Submit for App Review
- [ ] Monitor review status
- [ ] Address any rejection feedback
- [ ] Get approval ✅
- [ ] Release to App Store 🎉
- [ ] Monitor crash reports and reviews
- [ ] Plan version 1.1 improvements

📊 TESTING CHECKPOINTS SUMMARY

Throughout development, test the build at these key moments:

After Phase 1 (Core Foundation):

- ✅ Data persists correctly
- ✅ TMDb API successfully fetches movies
- ✅ Liquid Glass components render properly

After Phase 2 (Core Features):

- ✅ Can browse and view all movies
- ✅ Can rate and review movies
- ✅ Home screen displays stats and news

After Phase 3 (Advanced Features):

- ✅ Can rank movies with drag-and-drop
- ✅ Stats are accurate and visually appealing
- ✅ Custom tags work as expected

After Phase 4 (User Experience):

- ✅ Onboarding is smooth and helpful
- ✅ Settings allow full customization
- ✅ Search and filters work perfectly

After Phase 5 (Gamification & Social):

- ✅ Achievements unlock at right moments
- ✅ Can share content to social media
- ✅ Notifications work reliably

After Phase 6 (Widgets):

- ✅ All widget sizes display correctly
- ✅ Widgets update with app data

After Phase 7 (Export):

- ✅ Website exports successfully
- ✅ HTML is styled correctly

After Phase 8 (Polish):

- ✅ Animations are smooth
- ✅ App is fast and responsive
- ✅ VoiceOver works throughout

🎯 DEVELOPMENT BEST PRACTICES

Version Control

- Commit after each completed task
- Use descriptive commit messages

- Branch for experimental features
- Tag releases (v1.0, v1.1, etc.)

Code Quality

- Follow Swift style guidelines
- Use descriptive variable/function names
- Comment complex logic
- Keep files under 300 lines when possible
- Extract reusable components

Testing Strategy

- Write tests as you code (not after)
- Test on physical device regularly
- Test both iPhone and iPad layouts
- Test with poor network conditions
- Test with empty/populated data states

Performance Monitoring

- Profile with Instruments regularly
- Monitor memory usage
- Check for retain cycles
- Optimize before shipping

📝 NOTES & REMINDERS

Key Design Decisions Confirmed

- **App Name:** NCDB (Nicolas Cage Database)
- **Color Scheme:** Cage Gold (#FFD700), deep blacks, white text
- **Design Language:** Liquid Glass (frosted materials, depth, luminosity)
- **Target:** iOS 26+ (requires A13 Bionic or newer)
- **Architecture:** SwiftUI + SwiftData + MVVM
- **API:** TMDb for movie data
- **Preloaded Movies:** Face/Off, Con Air (bundled JSON)
- **Rating System:** 5-star with 0.5 increments

- **News Sources:** RSS feeds (Google News, IMDb, Variety filtered for Nic Cage)
- **Export Options:** Static HTML + FTP, or Strapi/Netlify
- **Actor Support:** Nicolas Cage primary, can add other actors later

Features Confirmed In Scope

✅ Movie tracking (watched/unwatched)
✅ Star ratings and written reviews
✅ Interactive ranking carousel
✅ Comprehensive statistics dashboard
✅ Custom tags/collections
✅ News aggregation
✅ Achievements system (34 achievements)
✅ Social sharing templates
✅ iOS widgets (all sizes)
✅ Static website export
✅ Onboarding flow
✅ Settings & preferences
✅ Offline mode with caching
✅ Rewatch tracking
✅ External ratings display (IMDb, RT)

Future Enhancement Ideas (Post-Launch)

- Multiple actor support (beyond Nic Cage)
- watchOS companion app
- macOS version
- iCloud sync
- Family sharing
- Watch party mode
- Advanced statistics (ML-powered insights)
- Integration with other movie tracking services
- Community features (friend rankings)

🎬 FINAL THOUGHTS

This roadmap provides a structured path from design to deployment. Each phase builds on the previous, with clear testing checkpoints to ensure quality at every stage.

Estimated Total Development Time: 15 weeks (part-time) or 8 weeks (full-time)

Remember to:

- Test early and often
- Prioritize user experience
- Keep the Liquid Glass aesthetic consistent
- Have fun building this unique tribute to Nicolas Cage! 🤹

Good luck with development! 🚀

Document Version: 1.0 Last Updated: November 22, 2025 Project: NCDB - Nicolas Cage Database iOS App