SE4041 - Mobile Application Design & Development

ASSIGNMENT 02

Fernando W W R D

IT22361554

SUMMARY

This report documents the design, implementation, testing, and development process of two companion applications: MovieMatch (iOS) and WatchParty (tvOS). Both applications address the universal problem of decision paralysis in entertainment selection through innovative user experiences tailored to their respective platforms.

MovieMatch (iOS) enables users to discover movies through mood-based filtering and an intuitive swipe interface, recommending content after just five interactions. The application features group coordination capabilities for collaborative movie nights.

WatchParty (tvOS) provides a television-optimized discovery experience with curated suggestions, search functionality, and a conceptual QR code-based voting system for group decision-making.

Key Achievements:

- Two fully functional applications across different Apple platforms
- Advanced SwiftUI implementation with custom components
- MVVM architecture ensuring clean code organization
- MapKit integration demonstrating emerging technology use
- Cohesive visual design with platform-appropriate interactions
- Comprehensive documentation and code structure

1. DESIGN PHASE

1.1 Problem Definition

Core Problem Identified:
Users spend 20-30 minutes browsing streaming services without making decisions, experiencing what behavioral psychology terms "choice paralysis." For groups, this decision time extends to 45-60 minutes with frequent compromise dissatisfaction.

Target Audience:

- Primary: Adults 18-45 with active streaming subscriptions
- Secondary: Friend groups and families seeking collaborative entertainment
- Characteristics: Tech-savvy, multiple streaming services, frequent viewing habits

1.2 Design Philosophy

iOS (MovieMatch) - Individual Experience:

- Minimize decision time through progressive disclosure
- Leverage familiar interaction patterns (swipe gestures)
- Prioritize emotional state over technical categorization
- Enable social coordination without complexity

tvOS (WatchParty) - Shared Experience:

- Optimize for distance viewing (10-foot UI)
- Support remote-based navigation and Focus Engine
- Facilitate group participation without friction
- Maintain visual consistency with iOS counterpart

1.3 Information Architecture

MovieMatch Navigation Structure:

```
Authentication Gate
  └── Main Tab View
    ├── Mood Selection Tab
    │  └── Mood → Swipe Handoff
    ├── Swipe Tab
    │  ├── Card Stack
    │  └── Suggestion Sheet (after 5 swipes)
    ├── Groups Tab
    │  ├── Map View (MapKit)
    │  ├── Group List
    │  └── Add/Edit Group Sheet
    └── Profile Tab
      ├── User Information
      └── Sign Out
```

WatchParty Navigation Structure:

```
Tab View
  ├── Home (Discovery Grid)
  ├── Search (Query Interface)
  ├── Player (Playback Mock)
  └── QR (Watch Party Join)
```

1.4 Visual Design System

Color Palette Rationale:

- **Black Background (#000000):** Cinematic feel, reduces eye strain, emphasizes content
- **Surface Color (#121212):** Differentiates cards while maintaining dark unity
- **Primary Red (#DC2626):** High contrast for calls-to-action, attention direction
- **White Text (#FFFFFF):** Maximum readability on dark backgrounds
- **Gray Text (#808080):** Hierarchical text differentiation

Typography:

- System fonts for platform consistency
- Large sizes for distance viewing (tvOS)
- Dynamic Type support structure (iOS)
- Clear hierarchy through weight and size

Component Design:

- Card-based interfaces for content focus
- Gradient overlays for text legibility
- Rounded corners (20pt) for modern aesthetic
- Consistent spacing (8pt grid system)

1.5 Interaction Design

iOS Gesture System:

- Horizontal drag for swipe decisions
- Threshold-based (100pt) for intentional actions
- Velocity consideration for natural feel
- Haptic feedback at decision points
- Fall-back button controls for accessibility

tvOS Focus System:

- Single-focus paradigm per screen
- Scale effects (1.0x → 1.1x) on focus
- Shadow intensification for depth perception
- Predictable focus flow patterns
- Remote-optimized target sizes

2. IMPLEMENTATION PHASE

2.1 Technology Stack

Development Environment:

- Xcode 15.0+
- Swift 5.9
- SwiftUI (declarative UI framework)
- Minimum iOS 16.0, tvOS 16.0

Architectural Pattern:

- MVVM (Model-View-ViewModel)
- Store pattern for persistence
- Service layer for external operations
- Environment objects for dependency injection

Key Frameworks:

- SwiftUI: User interface construction
- MapKit: Spatial data visualization
- Combine: Reactive programming
- Foundation: Core utilities

3. TESTING & QUALITY ASSURANCE

3.1 Testing Strategy

Development Testing:

- SwiftUI Previews: Rapid UI iteration and visual verification
- Xcode Simulator: Functional testing across device sizes
- Physical Device Testing: Performance and gesture validation

Manual Testing Checklist:

Feature	Test Case	Result
Authentication	Mock login/logout flow	✓ Pass
Mood Selection	All 7 moods selectable	✓ Pass
Mood-Swipe Handoff	Navigation + data transfer	✓ Pass
Swipe Gestures	Left/right recognition	✓ Pass
5-Swipe Suggestion	Triggers after exactly 5 swipes	✓ Pass
Group Creation	Add group with members	✓ Pass
Group Persistence	Data survives app restart	✓ Pass
Map Display	Pins render correctly	✓ Pass
tvOS Navigation	Remote control responsiveness	✓ Pass
tvOS Focus	Focus indicators visible	✓ Pass
tvOS Search	Query filtering works	✓ Pass

3.2 Code Quality Measures

Swift Best Practices Implemented:

- Guard statements for early exits
- Optional chaining and nil coalescing
- Computed properties for derived data
- Protocol conformance (Identifiable, Codable)
- Clear naming conventions
- Single Responsibility Principle

3.3 Performance Considerations

Optimization Techniques:

- LazyVGrid for efficient list rendering (tvOS)
- AsyncImage for non-blocking image loads
- Computed properties instead of stored calculations
- Minimal View re-renders through @Published granularity

Memory Management:

- Value types (structs) for models
- Weak references where appropriate
- Automatic reference counting (ARC) awareness

4. CHALLENGES & SOLUTIONS

4.1 Technical Challenges

Challenge 1: Swipe Gesture Conflicts

Problem:
Initial swipe implementation conflicted with ScrollView gestures, causing unintended scrolling during movie card swipes.

Solution:
Implemented gesture priority using `.highPriorityGesture()` modifier on card DragGesture, ensuring swipe takes precedence over scroll.

```
MovieCard(movie: movie)
    .gesture(DragGesture().onChanged { ... })
    .highPriorityGesture(swipeGesture)
```

Challenge 2: State Management Across Tabs

Problem:
Mood selection in MoodView needed to update SwipeView and trigger navigation, requiring state sharing across tab views.

Solution:
Created AppState as environment object with `selectedTab` published property, allowing MoodView to programmatically switch tabs.

```
class AppState: ObservableObject {
    @Published var selectedTab: Tab = .mood
}

// In MoodView
@EnvironmentObject var appState: AppState
Button("Select Mood") {
    appState.selectedTab = .swipe
}
```

Challenge 3: MapKit Annotation Customization

Problem:
Default MapPin appearance insufficient for conveying group information.

Solution:
While current implementation uses standard MapPin with custom tint, architecture supports MapAnnotation for fully custom views.

Challenge 4: tvOS Focus Debugging

Problem:
Focus state difficult to debug in Simulator; unclear which element had focus.

Solution:
Added explicit focus indicators (scale, shadow) and used `@Environment(\.isFocused)` to log focus changes during development.

4.2 Design Challenges

Challenge 1: Mood Category Definition

Problem:
Determining optimal number and naming of mood categories to avoid overwhelming users while providing meaningful filtering.

Solution:
Researched emotional psychology and streaming behavior, settled on 7 distinct, mutually exclusive moods covering primary emotional states.

Challenge 2: 5-Swipe Magic Number

Problem:
Determining appropriate number of swipes before suggestion.

Rationale:

- Too few (2-3): Insufficient data for pattern recognition
- Too many (10+): User fatigue, defeats quick decision goal
- Five: Sweet spot for pattern emergence without burden

Challenge 3: Dark Theme Accessibility

Problem:
Ensuring sufficient contrast ratios while maintaining aesthetic.

Solution:

- Tested color combinations against WCAG AA standards
- Pure white (#FFFFFF) on pure black (#000000) exceeds 21:1 ratio
- Red accent (#DC2626) maintains 4.5:1+ on dark backgrounds