Report Name: Oman Culture Mobile Application

Student ID: 2120246006 *Name:* Gheith ALRawahi

INTRODUCTION

This report presents the Oman Culture Mobile Application, a comprehensive Android app showcasing Oman's rich cultural heritage through famous Omani figures across different categories. The application is built using modern Android development technologies including Kotlin and Jetpack Compose.

Application Features:

- Multi-language support (English and Arabic with RTL layout)
- Interactive onboarding experience introducing Oman's heritage
- Browse famous Omani figures across 6 categories
- Detailed biographies and achievements
- Favorites system with local storage
- Modern Material 3 design
- Smooth animations and transitions
- Search functionality
- Category-based filtering

The app demonstrates best practices in Android development including MVVM architecture, clean code separation, state management with ViewModels, and efficient image loading. It features 20 carefully curated Omani figures spanning historical leaders, poets, artists, athletes, and scholars.

Source Code: The complete source code is available on GitHub at:
`https://github.com/gheith3/oman-culture-app`

SOLUTIONS

The application utilizes modern Android development tools and libraries:

2.1 Core Technologies

- **Kotlin**: Modern programming language for Android
- **Jetpack Compose**: Declarative UI framework for building native Android interfaces

- **Material 3**: Latest Material Design components and theming
- **Navigation Compose**: Type-safe navigation between screens
- **ViewModel & Lifecycle**: State management and lifecycle-aware components
- **Coil**: Efficient image loading library with caching support
- **DataStore Preferences**: Modern data storage for favorites
- **Compose Icons**: Material Icons Extended and Feather Icons

2.2 Architecture Pattern

The app follows **MVVM (Model-View-ViewModel)** architecture:

- **Model**: Data classes (Figure, Category) and Repository pattern
- **View**: Composable UI components and screens
- **ViewModel**: Business logic and state management
- **Repository**: Single source of truth for data access

2.3 Design System

The app uses a modern color palette with Material 3 design principles:

- **Primary Red (#E63946)**: Main accent color, headers, CTAs, gradient borders
- **Secondary Blue (#457B9D)**: Secondary accent, category colors
- **Tertiary Orange (#F4A261)**: Tertiary accent, highlights
- **Background**: Light (#FAFAFA) / Dark (#121212)
- **Surface**: White (#FFFFFF) / Dark Gray (#1E1E1E)

DETAILS

3.1 Project Structure

The application follows a clean, modular structure:

Listing 1: Project Directory Structure

```
app/src/main/java/com/oman/culture/
          MainActivity.kt
          OmanCultureApp.kt
          data/
                local/
                      FavoritesDataStore.kt
                model/
                      Figure.kt
                      Category.kt
```

```
                repository/
                    FiguresRepository.kt
        ui/
                theme/
                      Color.kt
                      Theme.kt
                      Type.kt
                      Shape.kt
                navigation/
                      NavGraph.kt
                components/
                      FigureCard.kt
                      FigureAvatar.kt
                      CategoryChip.kt
                      StatsRow.kt
                      AnimatedBottomBar.kt
                      LanguageSwitcher.kt
                      SearchBar.kt
                screens/
                    onboarding/
                          OnboardingScreen.kt
                          OnboardingPreferences.kt
                    home/
                          HomeScreen.kt
                          HomeViewModel.kt
                    detail/
                          DetailScreen.kt
                          DetailViewModel.kt
                    favorites/
                          FavoritesScreen.kt
                          FavoritesViewModel.kt
                    settings/
                        SettingsScreen.kt
        localization/
            LocaleManager.kt
```

3.2 Data Models

Figure Data Class:

The core data model supports bilingual content:

Listing 2: Figure.kt Data Model

```
data class Figure(
    val id: Int,
    val nameEn: String,
    val nameAr: String,
    val category: Category,
    val imageUrl: String,
    val descriptionEn: String,
    val descriptionAr: String,
    val biographyEn: String,
```

```
    val biographyAr: String,
    val achievementsEn: List<String>,
    val achievementsAr: List<String>,
    val era: String,
    val isFavorite: Boolean = false
) {
    fun getName(isArabic: Boolean) =
        if (isArabic) nameAr else nameEn
    fun getDescription(isArabic: Boolean) =
        if (isArabic) descriptionAr else descriptionEn
    fun getBiography(isArabic: Boolean) =
        if (isArabic) biographyAr else biographyEn
    fun getAchievements(isArabic: Boolean) =
        if (isArabic) achievementsAr else achievementsEn
}
```

Category Enum:
Six categories organize the figures:

Listing 3: Category.kt - Bilingual Enum

```
enum class Category(
    val id: String,
    val displayNameEn: String,
    val displayNameAr: String,
    val descriptionEn: String,
    val descriptionAr: String,
    val icon: ImageVector,
    val color: Color
) {
    HISTORICAL_LEADERS(
        id = "historical_leaders",
        displayNameEn = "Historical Leaders",
        displayNameAr = "                                    ",
        descriptionEn = "Sultans and rulers who shaped Oman's
           history",
        descriptionAr = "
                                                ",
        icon = Icons.Default.AccountBalance,
        color = Primary
    ),
    POETS_WRITERS(
        id = "poets_writers",
        displayNameEn = "Poets & Writers",
        displayNameAr = "                                  ",
        descriptionEn = "Literary figures who enriched Omani
           culture",
        descriptionAr = "
                                           ",
        icon = Icons.Default.MenuBook,
        color = Secondary
    );
    // ... other categories
```

```
    fun getDisplayName(isArabic: Boolean): String =
        if (isArabic) displayNameAr else displayNameEn

    fun getDescription(isArabic: Boolean): String =
        if (isArabic) descriptionAr else descriptionEn
}
```

3.3 User Interface Details

3.3.1 Onboarding Screen

A 4-page welcome flow introduces users to the app:

- **Page 1**: Welcome with Oman emblem
- **Page 2**: About Oman's heritage (5000+ years)
- **Page 3**: Preview of famous figures
- **Page 4**: Language selection (English/Arabic)

Features include horizontal paging, animated page indicators, skip functionality, and preference storage to show only on first launch.

3.3.2 Home Screen

The main screen displays:

- Header with app title and language switcher
- Search bar for finding figures
- Horizontal scrolling category filter chips
- Grid layout of figure cards with:
  - Circular avatar with gradient border (Red to Green)
  - Figure name (bilingual)
  - Category tag
  - Stats row (Era, Works count)
  - Favorite heart button
- Animated bottom navigation bar

3.3.3 Detail Screen

Comprehensive figure information:

- Large circular avatar with gradient ring
- Name in both English and Arabic
- Category badge with icon
- Stats row (Era — Works — Years Active)
- Tab navigation: Biography and Achievements
- Biography tab with full text content
- Achievements tab with bullet-point list
- Favorite toggle button
- Share functionality
- Back navigation

3.3.4 Favorites Screen

Manages saved figures:

- Header with count badge
- List of favorite figure cards
- Swipe-to-delete gesture with red background
- Empty state with animated heart icon
- "Explore Figures" call-to-action button
- Tap to navigate to detail view

3.3.5 Settings Screen

Configuration options:

- Language selection (English/Arabic)
- Theme toggle (Light/Dark mode)
- About section with app information
- App version display

3.4 User Operation Details

First Launch Flow:

1. User opens app for the first time
2. Onboarding screens appear
3. User swipes through 4 pages or taps "Skip"
4. User selects preferred language
5. Taps "Start Exploring" to enter main app
6. Onboarding preference saved (won't show again)

Main Usage Flow:

1. **Browse**: View all figures on home screen
2. **Filter**: Tap category chips to filter by category
3. **Search**: Type in search bar to find specific figures
4. **View Details**: Tap any card to see full biography
5. **Favorite**: Tap heart icon to save to favorites
6. **Navigate**: Use bottom bar to switch between sections
7. **Switch Language**: Tap language button in header

Favorites Management:

1. Tap heart icon on any figure card or detail screen
2. Navigate to Favorites tab via bottom navigation
3. View all saved figures
4. Swipe left on any card to remove from favorites
5. Tap card to view full details

3.5 Data Repository Details

FiguresRepository:
Provides centralized data access:

Listing 4: Repository Functions

```
class FiguresRepository {
    // Get all figures
    fun getAllFigures(): List<Figure>

    // Get single figure by ID
    fun getFigureById(id: Int): Figure?
```

```
    // Filter by category
    fun getFiguresByCategory(category: Category): List<Figure>

    // Get multiple figures by IDs (for favorites)
    fun getFiguresByIds(ids: List<Int>): List<Figure>

    // Search figures by name
    fun searchFigures(query: String): List<Figure>
}
```

The repository contains 20 carefully researched Omani figures including Sultan Qaboos, Ahmad bin Majid, Mazoon Al Alawi, and many others across all six categories.

3.6 Local Storage Details

FavoritesDataStore:

Uses Jetpack DataStore Preferences for persistent storage:

Listing 5: FavoritesDataStore.kt - Persistent Storage

```
private val Context.dataStore: DataStore<Preferences>
    by preferencesDataStore(name = "favorites")

class FavoritesDataStore(private val context: Context) {

    private val favoriteIdsKey = stringSetPreferencesKey("
       favorite_ids")

    val favoriteIds: Flow<Set<Int>> = context.dataStore.data.map
       { preferences ->
        preferences[favoriteIdsKey]?.mapNotNull { it.toIntOrNull
           () }?.toSet()
            ?: emptySet()
    }

    suspend fun addFavorite(figureId: Int) {
        context.dataStore.edit { preferences ->
            val currentFavorites = preferences[favoriteIdsKey] ?:
                emptySet()
            preferences[favoriteIdsKey] = currentFavorites +
               figureId.toString()
        }
    }

    suspend fun removeFavorite(figureId: Int) {
        context.dataStore.edit { preferences ->
            val currentFavorites = preferences[favoriteIdsKey] ?:
                emptySet()
            preferences[favoriteIdsKey] = currentFavorites -
               figureId.toString()
        }
    }
```

```
    suspend fun toggleFavorite(figureId: Int) {
        context.dataStore.edit { preferences ->
            val currentFavorites = preferences[favoriteIdsKey] ?:
                emptySet()
            val figureIdStr = figureId.toString()
            preferences[favoriteIdsKey] = if (figureIdStr in
               currentFavorites) {
                currentFavorites - figureIdStr
            } else {
                currentFavorites + figureIdStr
            }
        }
    }

    fun isFavorite(figureId: Int): Flow<Boolean> =
        context.dataStore.data.map { preferences ->
            val favorites = preferences[favoriteIdsKey] ?:
               emptySet()
            figureId.toString() in favorites
        }
}
```

Data persists across app restarts and is automatically synchronized with the UI through Kotlin Flows.

3.7 Localization Details

Multi-Language Support:

- **String Resources**: Separate XML files for English (values/) and Arabic (values-ar/)
- **RTL Layout**: Automatic right-to-left layout for Arabic
- **LocaleManager**: Manages language switching and persistence
- **Bilingual Content**: All figures have English and Arabic names, descriptions, biographies, and achievements
- **Auto-mirrored Icons**: Directional icons automatically flip for RTL

Locale Configuration:

Listing 6: LocaleManager.kt

```
class LocaleManager(private val context: Context) {

    private val prefs = context.getSharedPreferences(PREFS_NAME,
       Context.MODE_PRIVATE)

    fun getLanguage(): String {
        return prefs.getString(KEY_LANGUAGE, DEFAULT_LANGUAGE) ?:
            DEFAULT_LANGUAGE
```

```
    }

    fun setLanguage(language: String) {
        prefs.edit().putString(KEY_LANGUAGE, language).apply()
    }

    fun isArabic(): Boolean = getLanguage() == ARABIC

    fun setLocale(context: Context): Context {
        return updateResources(context, getLanguage())
    }

    private fun updateResources(context: Context, language:
       String): Context {
        val locale = Locale(language)
        Locale.setDefault(locale)

        val config = Configuration(context.resources.
           configuration)
        config.setLocale(locale)
        config.setLayoutDirection(locale)

        return context.createConfigurationContext(config)
    }

    companion object {
        private const val PREFS_NAME = "locale_prefs"
        private const val KEY_LANGUAGE = "selected_language"
        private const val DEFAULT_LANGUAGE = "en"
        const val ENGLISH = "en"
        const val ARABIC = "ar"
    }
}
```

3.8 Animation Details

Implemented Animations:

- **Card Entrance**: Staggered fade-in and slide-up animation
- **Card Press**: Scale animation on tap
- **Favorite Heart**: Pulse animation when toggled
- **Avatar Rotation**: Optional infinite rotation animation for gradient border
- **Bottom Bar**: Animated indicator sliding between tabs
- **Screen Transitions**: Fade and slide between screens
- **Tab Content**: Crossfade when switching tabs

- **Swipe to Delete**: Red background reveal with shrink animation
- **Empty State**: Infinite pulse animation on heart icon
- **Page Indicator**: Animated dots on onboarding

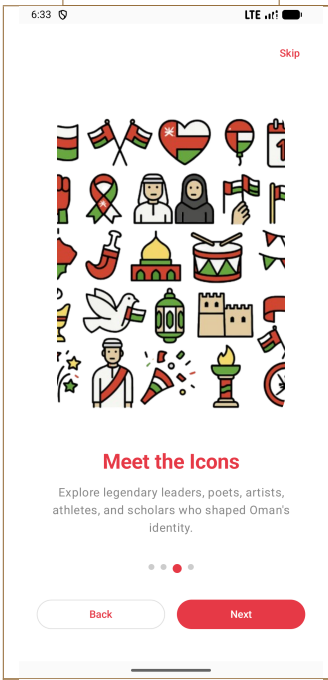
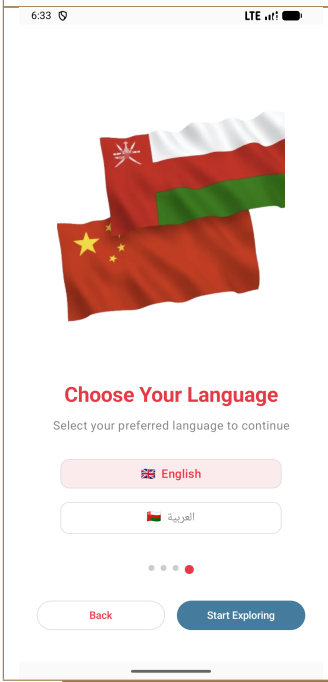
SCREENSHOTS

Table 1: Application Screenshots with Descriptions

Screenshots	Descriptions
	**Onboarding - Welcome:** First page showing Oman map with national flag colors and emblem. Title "Welcome to Oman Culture" with subtitle "Discover the rich heritage of the Sultanate of Oman". Page indicator dots (1 of 4) and Next button.
	**Onboarding - Land of Heritage:** Second page featuring illustration of traditional Omani fort. Describes Oman's 5,000 years of civilization, ancient forts, and warm hospitality. Back and Next navigation buttons.

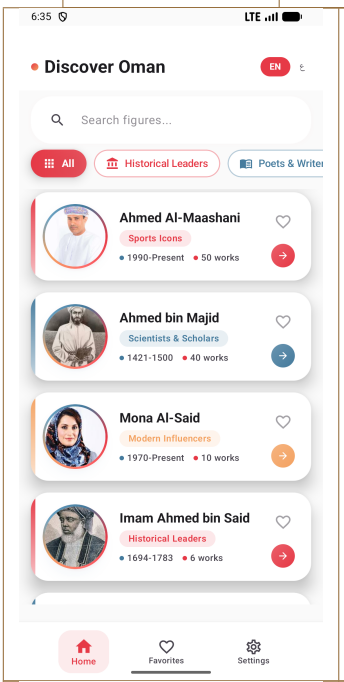
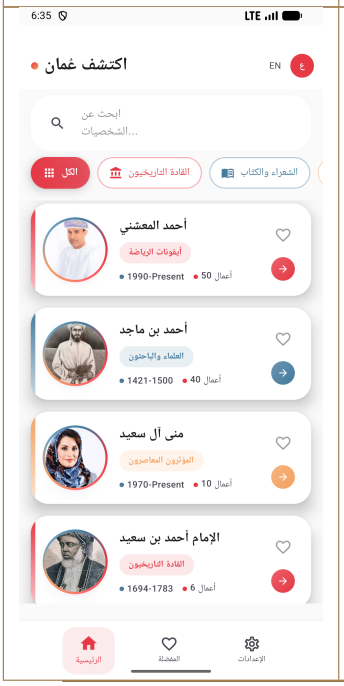
Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Onboarding - Meet the Icons:** Third page with Omani cultural icons collage (traditional dress, mosque, lantern, fort, flags). Introduces the app's purpose: exploring legendary leaders, poets, artists, athletes, and scholars.
	**Onboarding - Language Selection:** Final page with Oman flag imagery. Two language options: English (UK flag) and /Arabic (Oman flag). "Start Exploring" button to enter the main app.

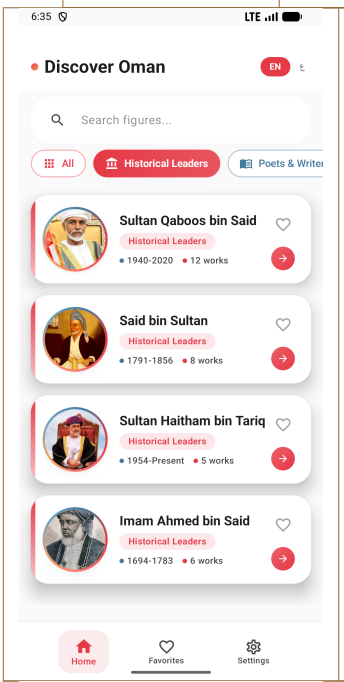
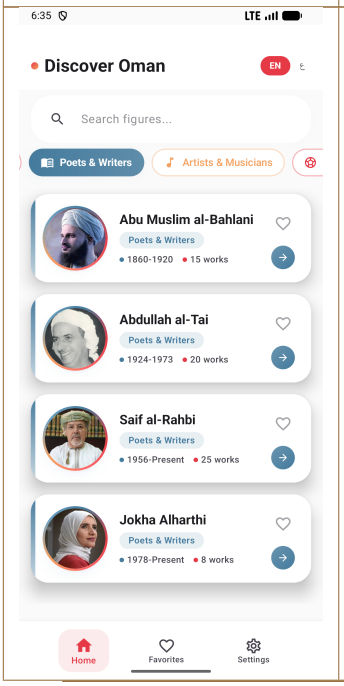
Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Home Screen (English):** Main screen with "Discover Oman" header and EN/AR language toggle. Search bar placeholder "Search figures...". Category chips: All, Historical Leaders, Poets & Writers. Figure cards showing Ahmed Al-Maashani, Ahmed bin Majid, Mona Al-Said, and Imam Ahmed bin Said with gradient avatar borders.
	**Home Screen (Arabic - RTL):** *Multi-language feature demonstration.* Same screen in Arabic with complete RTL layout. Header shows " ". All figure names in Arabic ( ). Category chips and bottom navigation labels translated. Demonstrates automatic layout direction reversal.

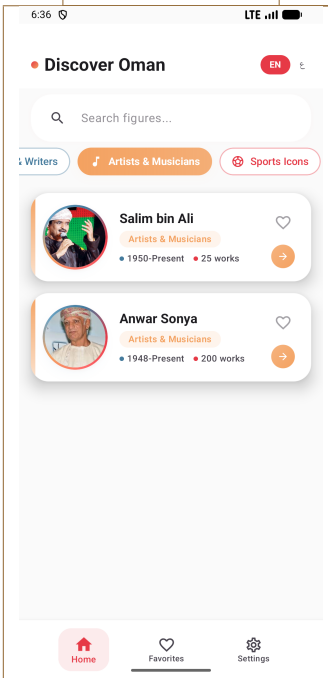
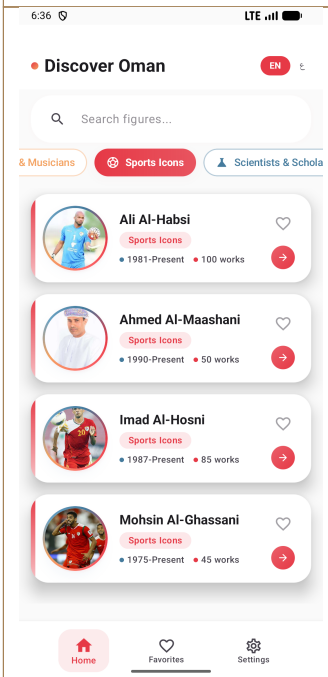
Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Category Filter - Historical Leaders:** Filtered view showing only Historical Leaders category (red chip selected). Displays Sultan Qaboos bin Said (1940-2020), Said bin Sultan (1791-1856), Sultan Haitham bin Tariq, and Imam Ahmed bin Said.
	**Category Filter - Poets & Writers:** Blue chip selected showing literary figures: Abu Muslim al-Bahlani (1860-1920), Abdullah al-Tai (1924-1973), Saif al-Rahbi (1956-Present), and Jokha Alharthi (1978-Present, Man Booker Prize winner).

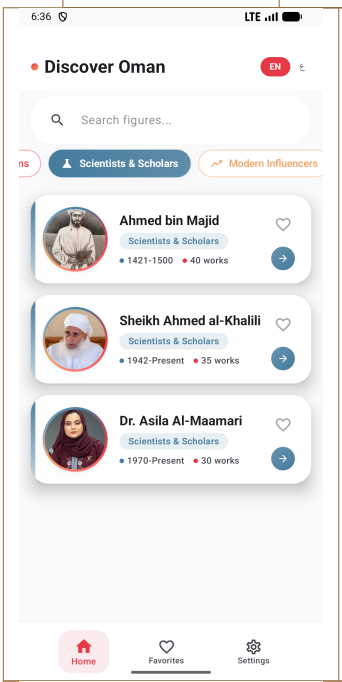
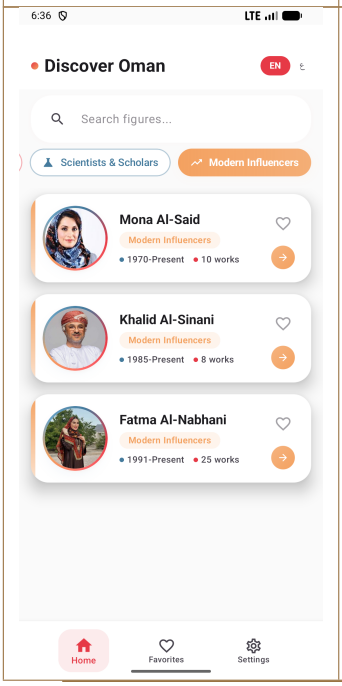
Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Category Filter - Artists & Musicians:** Orange chip selected displaying artists: Salim bin Ali (1950-Present, 25 works) and Anwar Sonya (1948-Present, 200 works). Shows category-specific color coding.
	**Category Filter - Sports Icons:** Red chip selected showing athletes: Ali Al-Habsi (goalkeeper, 1981-Present, 100 works), Ahmed Al-Maashani, Imad Al-Hosni (1987-Present), and Mohsin Al-Ghassani (1975-Present).

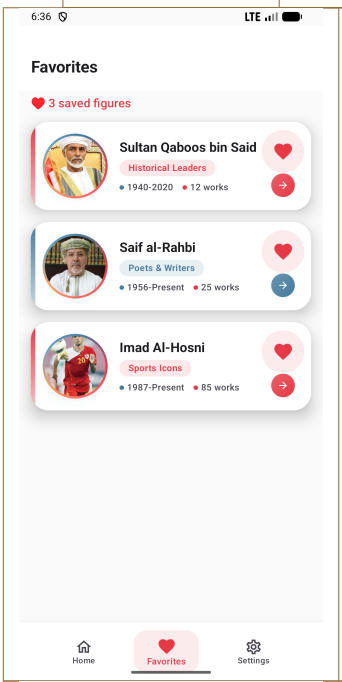
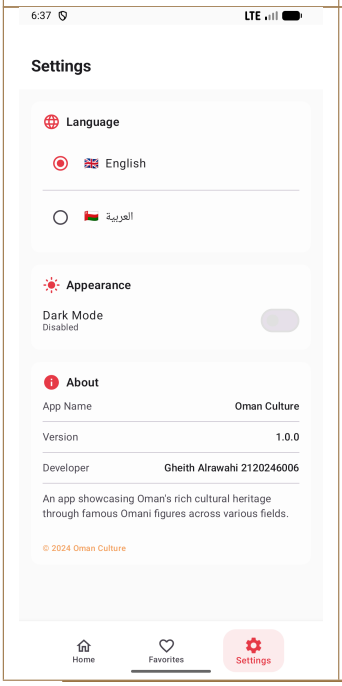
Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Category Filter - Scientists & Scholars:** Blue chip selected showing: Ahmed bin Majid (famous navigator, 1421-1500, 40 works), Sheikh Ahmed al-Khalili (1942-Present, 35 works), and Dr. Asila Al-Maamari (1970-Present, 30 works).
	**Category Filter - Modern Influencers:** Orange chip selected displaying contemporary figures: Mona Al-Said (1970-Present, 10 works), Khalid Al-Sinani (1985-Present, 8 works), and Fatma Al-Nabhani (tennis player).

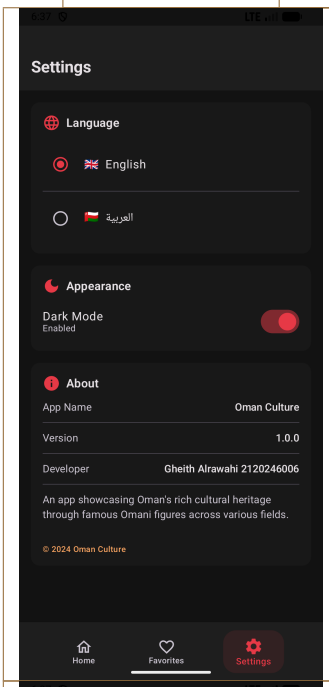
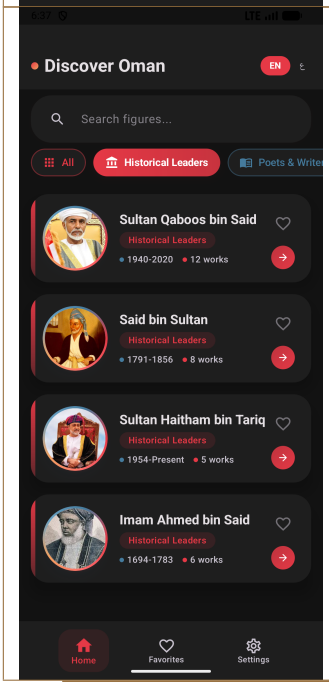
Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Favorites Screen:** Shows "Favorites" header with "3 saved figures" count. Lists Sultan Qaboos bin Said (Historical Leaders), Saif al-Rahbi (Poets & Writers), and Imad Al-Hosni (Sports Icons). Each card has filled red heart icon and arrow button.
	**Settings Screen (Light Mode):** Configuration page with Language section (English selected, Arabic option), Appearance section (Dark Mode toggle disabled), and About section showing App Name: Oman Culture, Version: 1.0.0, Developer: Gheith Alrawahi 2120246006.

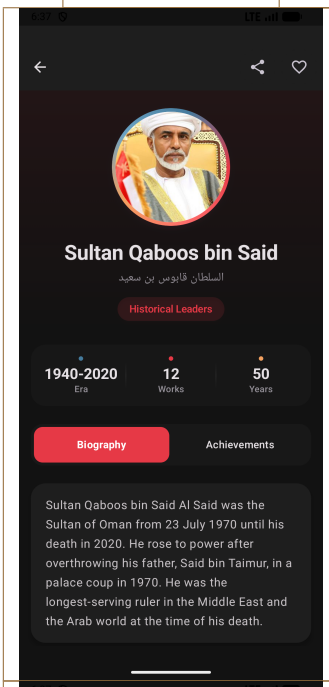
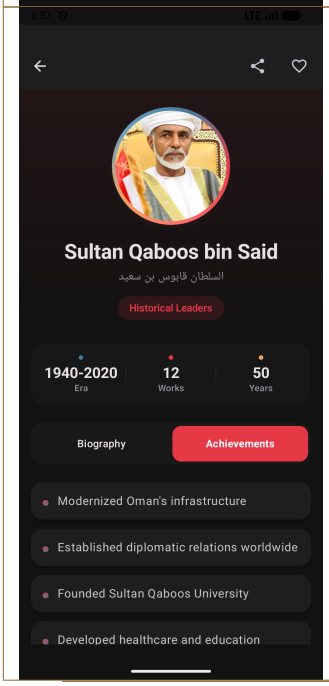
Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Settings Screen (Dark Mode):** *Theme switching demonstration.* Same settings page with Dark Mode enabled. Dark background with light text. Shows theme toggle in "Enabled" state. Demonstrates app-wide dark theme support.
	**Home Screen (Dark Mode):** *Theme feature demonstration.* Dark theme applied to home screen. Dark background with light text and cards. Category chips and figure cards adapt to dark color scheme while maintaining readability.

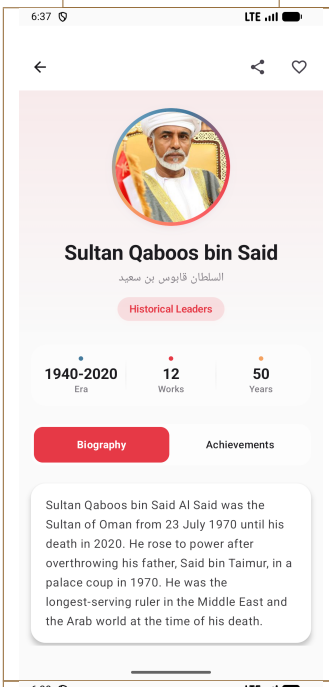
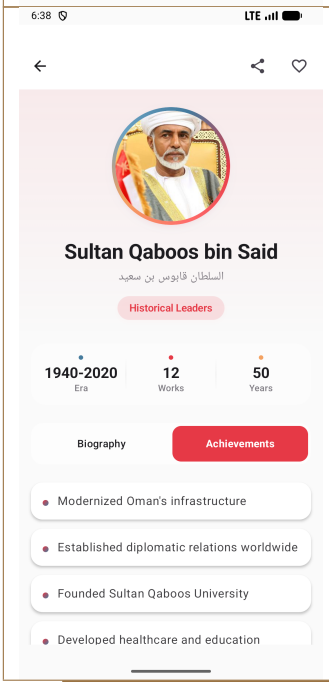
Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Detail Screen - Biography (Dark Mode):** Sultan Qaboos profile in dark theme. Large circular avatar with gradient border. Name in English and Arabic ( ). Stats row: Era 1940-2020, Works 12, Years 50. Biography tab selected with full text content.
	**Detail Screen - Achievements (Dark Mode):** Same profile with Achievements tab selected. Bullet-point list: Modernized Oman's infrastructure, Established diplomatic relations worldwide, Founded Sultan Qaboos University, Developed healthcare and education.

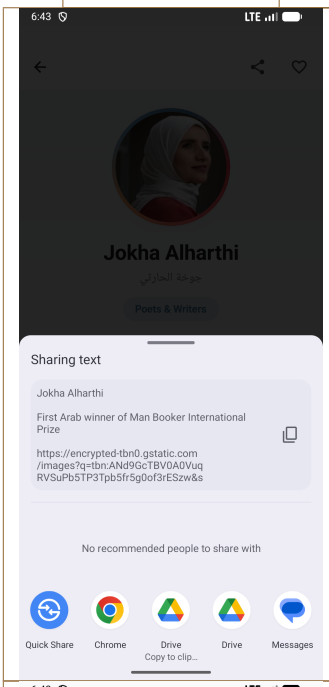
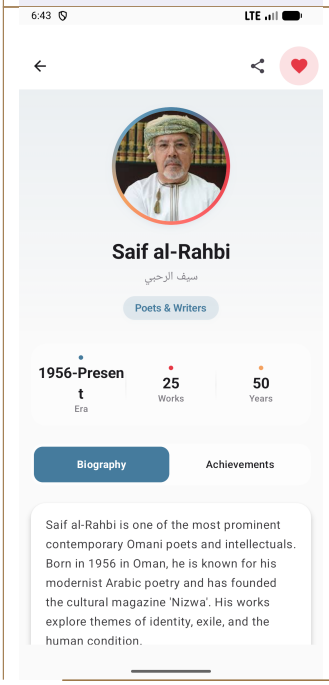
Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Detail Screen - Biography (Light Mode):** *Comparison with dark mode.* Same Sultan Qaboos profile in light theme. Shows consistent layout and information across themes. Pink gradient header, white content area.
	**Detail Screen - Achievements (Light Mode):** Light theme achievements view. Same content as dark mode version, demonstrating theme consistency. Clean white background with colored accent elements.

Continued on next page

Table 1 – *Continued from previous page*

Screenshots	Descriptions
	**Share Feature:** Share dialog for Jokha Alharthi (Poets & Writers). Shows sharing text with name, description "First Arab winner of Man Booker International Prize", and image URL. Share options: Quick Share, Chrome, Drive, Messages.
	**Detail Screen with Favorite:** Saif al-Rahbi profile (Poets & Writers, 1956-Present). Shows filled red heart icon indicating figure is saved to favorites. Biography describes him as prominent contemporary Omani poet and founder of 'Nizwa' cultural magazine.

CODE IMPLEMENTATION

5.1 Theme Configuration

Listing 7: Color.kt - Theme Colors

```
// Primary
val Primary = Color(0xFFE63946)
val PrimaryLight = Color(0xFFFF6B6B)
val PrimaryDark = Color(0xFFB82E3B)
```

```
// Secondary
val Secondary = Color(0xFF457B9D)
val SecondaryLight = Color(0xFF6BA3C4)
val SecondaryDark = Color(0xFF1D3557)

// Tertiary
val Tertiary = Color(0xFFF4A261)

// Background & Surface
val BackgroundLight = Color(0xFFFAFAFA)
val BackgroundDark = Color(0xFF121212)
val SurfaceLight = Color(0xFFFFFFFF)
val SurfaceDark = Color(0xFF1E1E1E)

private val LightColorScheme = lightColorScheme(
    primary = Primary,
    onPrimary = Color.White,
    primaryContainer = PrimaryLight,
    secondary = Secondary,
    tertiary = Tertiary,
    background = BackgroundLight,
    surface = SurfaceLight,
    onBackground = Color(0xFF1C1B1F),
    onSurface = Color(0xFF1C1B1F)
)
```

5.2 Navigation Setup

Listing 8: NavGraph.kt - Navigation with Animations

```
@Composable
fun NavGraph(
    navController: NavHostController,
    startDestination: String,
    isArabic: Boolean = false,
    favoriteIds: Set<Int> = emptySet(),
    onToggleFavorite: (Int) -> Unit = {}
) {
    NavHost(
        navController = navController,
        startDestination = startDestination
    ) {
        // Onboarding with fade animation
        composable(
            route = Screen.Onboarding.route,
            enterTransition = { fadeIn(tween(300)) },
            exitTransition = { fadeOut(tween(300)) }
        ) {
            OnboardingScreen(
                onComplete = {
                    navController.navigate(Screen.Home.route) {
```

```
                        popUpTo(Screen.Onboarding.route) {
                            inclusive = true }
                    }
                }
            )
        }

        // Detail with slide animation
        composable(
            route = Screen.Detail.route,
            arguments = listOf(
                navArgument("figureId") { type = NavType.IntType
                    }
            ),
            enterTransition = {
                slideIntoContainer(
                    towards = SlideDirection.Left,
                    animationSpec = tween(300)
                )
            },
            exitTransition = {
                slideOutOfContainer(
                    towards = SlideDirection.Right,
                    animationSpec = tween(300)
                )
            }
        ) { backStackEntry ->
            val figureId = backStackEntry.arguments?.getInt("
                figureId") ?: 0
            DetailScreen(
                figureId = figureId,
                isArabic = isArabic,
                isFavorite = figureId in favoriteIds,
                onBackClick = { navController.popBackStack() },
                onToggleFavorite = { onToggleFavorite(figureId) }
            )
        }
    }
}
```

5.3 Key Components

Listing 9: FigureAvatar.kt - Animated Gradient Avatar

```
@Composable
fun FigureAvatar(
    imageUrl: String,
    contentDescription: String?,
    modifier: Modifier = Modifier,
    size: Dp = 120.dp,
    borderWidth: Dp = 4.dp,
```

```
    animated: Boolean = false
) {
    val infiniteTransition = rememberInfiniteTransition()
    val rotation by infiniteTransition.animateFloat(
        initialValue = 0f,
        targetValue = 360f,
        animationSpec = infiniteRepeatable(
            animation = tween(8000, easing = LinearEasing),
            repeatMode = RepeatMode.Restart
        )
    )

    Box(
        modifier = modifier
            .size(size)
            .shadow(elevation = 8.dp, shape = CircleShape)
    ) {
        // Sweep gradient border with optional rotation
        Box(
            modifier = Modifier
                .fillMaxSize()
                .then(if (animated) Modifier.rotate(rotation)
                   else Modifier)
                .background(
                    brush = Brush.sweepGradient(
                        colors = listOf(
                            Primary, PrimaryLight, Tertiary,
                            SecondaryLight, Secondary, Primary
                        )
                    ),
                    shape = CircleShape
                )
                .padding(borderWidth)
        ) {
            SubcomposeAsyncImage(
                model = ImageRequest.Builder(LocalContext.current
                   )
                    .data(imageUrl)
                    .crossfade(true)
                    .build(),
                contentDescription = contentDescription,
                modifier = Modifier.fillMaxSize().clip(
                   CircleShape),
                contentScale = ContentScale.Crop,
                loading = {
                    Icon(
                        imageVector = Icons.Default.Person,
                        contentDescription = null,
                        tint = Primary.copy(alpha = 0.3f)
                    )
                }
```

```
            )
        }
    }
}
```

5.4 ViewModel Example

Listing 10: HomeViewModel.kt - State Management

```
data class HomeUiState(
    val figures: List<Figure> = emptyList(),
    val filteredFigures: List<Figure> = emptyList(),
    val categories: List<Category> = emptyList(),
    val selectedCategory: Category? = null,
    val searchQuery: String = "",
    val isLoading: Boolean = true,
    val favoriteIds: Set<Int> = emptySet(),
    val isArabic: Boolean = false
)

class HomeViewModel(application: Application) : AndroidViewModel(
   application) {

    private val repository = FiguresRepository(application.
       applicationContext)
    private val _uiState = MutableStateFlow(HomeUiState())
    val uiState: StateFlow<HomeUiState> = _uiState.asStateFlow()

    init {
        loadData()
    }

    private fun loadData() {
        viewModelScope.launch {
            val figures = repository.getAllFigures()
            val categories = repository.getCategories()
            _uiState.update {
                it.copy(
                    figures = figures,
                    filteredFigures = figures,
                    categories = categories,
                    isLoading = false
                )
            }
        }
    }

    fun onSearchQueryChange(query: String) {
        _uiState.update { state ->
            val filtered = if (query.isBlank()) {
```

```
                applyFilters(state.figures, state.
                    selectedCategory)
            } else {
                repository.searchFigures(query).let {
                    searchResults ->
                     applyFilters(searchResults, state.
                         selectedCategory)
                }
            }
            state.copy(searchQuery = query, filteredFigures =
                filtered)
        }
    }

    fun onCategorySelected(category: Category?) {
        _uiState.update { state ->
            val filtered = applyFilters(state.figures, category)
            state.copy(selectedCategory = category,
                filteredFigures = filtered)
        }
    }

    private fun applyFilters(figures: List<Figure>, category:
        Category?): List<Figure> {
        return if (category == null) figures
                else figures.filter { it.category == category }
    }
}
```

TESTING & QUALITY ASSURANCE

6.1 Testing Performed

- **Navigation Testing**: All screen transitions work correctly
- **Language Switching**: Instant language change without restart
- **RTL Layout**: Complete layout reversal for Arabic verified
- **Favorites Persistence**: Data survives app restart
- **Search Functionality**: Real-time filtering works correctly
- **Category Filtering**: Proper figure filtering by category
- **Animations**: Smooth 60fps animations throughout
- **Image Loading**: Efficient caching with Coil
- **Screen Rotation**: State preservation on rotation
- **Empty States**: Proper handling of no results/favorites

6.2 Performance Optimizations

- **LazyColumn/LazyVerticalGrid**: Efficient list rendering
- **Image Caching**: Coil memory and disk caching enabled
- **State Hoisting**: Minimized recompositions
- **Remember & Keys**: Proper state management in composables
- **R8 Minification**: Code shrinking and optimization enabled
- **Coroutines**: Asynchronous operations for smooth UI

CONCLUSIONS

The Oman Culture Mobile Application successfully demonstrates modern Android development practices while celebrating Oman's rich cultural heritage. The app achieves its goals of providing an engaging, educational, and beautifully designed platform for exploring famous Omani figures.

Key Achievements:

- **Complete Implementation**: All planned features successfully implemented
- **Modern Architecture**: Clean MVVM architecture with proper separation of concerns
- **Bilingual Support**: Full English and Arabic localization with RTL layout
- **Rich Content**: 20 carefully researched Omani figures across 6 categories
- **Smooth UX**: Polished animations and transitions throughout
- **Persistent Storage**: Favorites system with DataStore
- **Material Design**: Consistent use of Material 3 components
- **Modern Color Scheme**: Red, blue, and orange palette with gradient effects

Technical Highlights:

- Jetpack Compose for declarative UI
- Navigation Compose for type-safe navigation
- ViewModel for lifecycle-aware state management
- Coil for efficient image loading
- DataStore for modern data persistence
- Kotlin Flows for reactive data streams
- Material 3 theming system
- Comprehensive localization support

Future Enhancement Opportunities:

- **Backend Integration**: Connect to REST API for dynamic content updates
- **User Accounts**: Add authentication and cloud sync for favorites
- **Social Features**: Share figures on social media platforms
- **Offline Mode**: Cache all content for offline access
- **Audio Content**: Add audio biographies and pronunciations
- **Quiz Feature**: Interactive quizzes about Omani culture
- **Timeline View**: Historical timeline of figures
- **AR Experience**: Augmented reality for landmarks and monuments
- **Accessibility**: Enhanced support for screen readers and TalkBack
- **Widget Support**: Home screen widgets for daily figure highlights

Learning Outcomes:

- Mastery of Jetpack Compose declarative UI framework
- Implementation of MVVM architecture pattern
- Multi-language app development with RTL support
- State management with ViewModels and Flows
- Modern Android navigation patterns
- Material Design 3 theming and components
- Local data persistence with DataStore
- Image loading and caching strategies
- Animation and transition implementation
- Clean code architecture and best practices

The Oman Culture app serves as both a functional educational tool and a demonstration of modern Android development capabilities. It successfully combines technical excellence with cultural appreciation, creating an engaging platform for users to discover and learn about Oman's remarkable heritage and its influential figures throughout history.