ReadyRun 상세기획서 v3.0

마라톤 대회 정보 앱 - 프로덕트 레벨 기획서

📱 UI/UX 상세 설계

4.1 디자인 시스템

Color Palette

```
Primary Colors:
- Blue: #007AFF (iOS System Blue)
- Orange: #FF6B35 (CTA, Marathon highlights)
- Green: #34C759 (Success, Available)
- Amber: #FF9500 (Warning, Deadline alerts)
- Red: #FF3B30 (Error, Expired)

Neutral Colors:
- Background: #F2F2F7 (iOS System Background)
- Text Primary: #000000
- Text Secondary: #6C6C70
- Card Background: #FFFFFF
- Separator: #E5E5EA
```

Typography

```
- Display: SF Pro Display (32pt, Bold) - Hero titles
- Headline: SF Pro Text (22pt, Semibold) - Section headers
- Body: SF Pro Text (17pt, Regular) - Main content
- Caption: SF Pro Text (13pt, Regular) - Metadata
- Small: SF Pro Text (11pt, Regular) - Fine print
```

Spacing System

```
- XXS: 4pt
- XS: 8pt
- SM: 12pt
- MD: 16pt
- LG: 24pt
- XL: 32pt
- XXL: 48pt
```

Component Library

```
- Marathon Card: 335x180pt with rounded corners (12pt)
- Action Button: Height 50pt, Corner radius 25pt
- Search Bar: Height 44pt, Corner radius 22pt
- Filter Chip: Height 32pt, Corner radius 16pt
- Section Header: Height 44pt with "See All" link
```

4.2 화면별 상세 UI 명세

4.2.1 홈 화면 (Home Screen)

```
Navigation Bar:
├── Title: "ReadyRun" (SF Pro Display, 22pt, Bold)
├── Location Icon + Current City (Tap to change)
└── Notification Bell (Badge indicator)

Search Bar:
├── Placeholder: "마라톤 대회 검색..."
├── Search Icon (SF Symbol: magnifyingglass)
└── Voice Search Icon (SF Symbol: mic.fill)

Content Sections:
├── "내 주변 대회" Section
│   ├── Section Header with "모두 보기" link
│   ├── Marathon Cards (Vertical scroll, 2-3 visible)
│   └── Distance indicator (from user location)
│
├── "인기 대회" Section
│   ├── Section Header with "모두 보기" link
│   ├── Horizontal scroll cards (4-5 visible)
│   └── Popularity indicator (favorite count)
│
├── "마감 임박" Section
│   ├── Section Header with countdown badge
│   ├── Urgent-style cards (red accent)
│   └── Days remaining indicator
│
└── "추천 대회" Section
    ├── AI-based recommendations
    ├── User preference matching
    └── "왜 추천?" explanation text

Bottom Components:
├── AdMob Banner (320x50pt)
└── Tab Bar Navigation
```

4.2.2 검색 화면 (Search Screen)

```
Search Header:
├── Search Input Field
│   ├── Real-time search suggestions
│   ├── Search history (최근 검색어)
│   └── Auto-complete functionality
├── Filter Button (Badge with active filter count)
└── Cancel/Clear button

Active Filters Bar:
├── Scrollable filter chips
├── Remove filter "X" buttons
└── Clear all filters option

Sort Options:
├── Dropdown/Picker
├── Options: 날짜순, 인기순, 거리순, 가격순
└── Ascending/Descending toggle

Results List:
├── Marathon Cards (Compact view)
│   ├── Thumbnail image (80x80pt)
│   ├── Title, Date, Location
│   ├── Distance badges
│   ├── Price and currency
│   ├── Favorite button
│   └── View count indicator
├── Infinite scroll pagination
├── "Load more" indicator
└── Empty state (No results found)

Filter Modal:
├── Country/Region selector
├── Date range picker
├── Distance checkboxes (5K, 10K, Half, Full)
├── Price range slider
├── Tags selection (multi-select)
├── Difficulty level selector
└── Apply/Reset buttons
```

4.2.3 대회 상세 화면 (Marathon Detail)

```
Header:
├── Back button with title
├── Marathon name
├── Share button
└── Favorite toggle (heart icon)

Hero Section:
├── Hero image with overlay
├── Image gallery indicator (1/5)
├── Swipe gesture for multiple images
└── Full-screen image viewer

Basic Info Card:
├── Date and time (with calendar icon)
├── Location with map pin
├── Available distances (badge style)
├── Registration fee and currency
├── Deadline with countdown
├── Participant count/limit
└── Registration status indicator

Action Buttons:
├── Primary: "등록하기" (Full width)
├── Secondary: "길찾기" (Opens Maps app)
├── Tertiary: "공유하기" (Native share sheet)
└── "공식 웹사이트" link

Details Sections:
├── Description (Expandable text)
├── Course Map (Interactive/Static image)
├── Elevation Profile (Chart)
├── Weather Forecast (If within 7 days)
├── Organizer Information
├── Past Results/Statistics
├── Similar Events
└── User Reviews/Ratings

Floating Elements:
├── Back to top button (after scroll)
├── Quick action sheet (bottom swipe)
└── AdMob Banner (between sections)
```

4.2.4 즐겨찾기 화면 (Favorites)

```
Header:
├── Title: "즐겨찾기"
├── Edit button (bulk actions)
└── Sort options

Content Sections:
├── "다가오는 대회" (Upcoming)
│   ├── Countdown timers
│   ├── Registration status alerts
│   └── Notification settings per event
│
├── "등록한 대회" (Registered)
│   ├── QR codes for check-in
│   ├── Training tips
│   └── Race day information
│
└── "지난 대회" (Past Events)
    ├── Results if available
    ├── Photo uploads
    └── Review/Rating options

List Management:
├── Swipe to delete
├── Bulk select mode
├── Export to calendar
└── Share list functionality

Empty States:
├── No favorites illustration
├── Motivational message
└── "둘러보기" CTA button
```

4.2.5 프로필 화면 (Profile)

```
Profile Header:
├── Avatar (editable)
├── Display name
├── Running stats summary
└── Achievement badges

Settings Sections:
├── Account Settings
│   ├── Edit profile
│   ├── Email preferences
│   └── Privacy settings
│
├── App Preferences
│   ├── Language selection
│   ├── Units (Metric/Imperial)
│   ├── Preferred distances
│   ├── Notification settings
│   └── Location permissions
│
├── Data & Privacy
│   ├── Export data
│   ├── Delete account
│   └── Privacy policy
│
└── Support & Feedback
    ├── Help center
    ├── Contact support
    ├── Rate app
    └── Send feedback

Statistics Dashboard:
├── Total events favorited
├── Events attended
├── Countries visited
├── Total distance planned
└── Annual running goals
```

🎯 사용자 경험 (UX) 플로우

5.1 주요 사용자 여정 (User Journey)

5.1.1 신규 사용자 온보딩

```
Step 1: 스플래시 스크린
├── ReadyRun 로고 애니메이션
├── Loading indicator
└── 최소 2초 노출

Step 2: 웰컴 스크린
├── 앱 소개 메시지
├── 주요 기능 하이라이트
└── "시작하기" CTA

Step 3: 권한 요청 시퀀스
├── 위치 권한 (선택사항)
│   ├── 목적 설명
│   ├── 혜택 강조
│   └── 나중에 설정 옵션
├── 알림 권한 (선택사항)
│   ├── 알림 유형 설명
│   └── 맞춤 설정 제안
└── 개인정보 처리방침 동의

Step 4: 개인화 설정
├── 선호 거리 선택
├── 관심 지역 설정
├── 경험 수준 선택
└── 설정 완료 확인

Step 5: 메인 화면 진입
├── 튜토리얼 툴팁 (선택사항)
├── 첫 번째 추천 대회
└── 온보딩 완료
```

5.1.2 대회 검색 및 등록 플로우

```
검색 시작:
User opens app → Home screen → Tap search bar

검색 수행:
├── Type query OR Select filter
├── View results with sorting options
├── Tap on marathon card
└── Navigate to detail screen

상세 정보 확인:
├── Review marathon details
├── Check course map
├── View registration info
├── Add to favorites (optional)
└── Decide to register

등록 프로세스:
├── Tap "등록하기" button
├── Redirect to official website
├── Return to app (deep link)
├── Mark as "registered" in favorites
└── Set up notifications
```

5.1.3 즐겨찾기 관리 플로우

```
즐겨찾기 추가:
Marathon detail screen → Tap heart icon → Added confirmation

즐겨찾기 관리:
├── View favorites tab
├── Organize by categories
├── Set notification preferences
├── Remove completed events
└── Export to calendar
```

5.2 인터랙션 디자인

5.2.1 제스처 및 애니메이션

```
Navigation Gestures:
├── Swipe back (iOS standard)
├── Pull to refresh (list screens)
├── Infinite scroll (search results)
└── Long press (context menus)

Visual Feedback:
├── Button press animations (scale + haptic)
├── Loading states (skeleton screens)
├── Success/Error toast messages
├── Favorite heart animation
└── Smooth transitions (300ms ease-out)

Micro-interactions:
├── Search bar focus animation
├── Filter badge appearance
├── Card hover effects
├── Tab switching animation
└── Pull-to-refresh indicator
```

5.2.2 접근성 (Accessibility)

```
VoiceOver Support:
├── Meaningful accessibility labels
├── Navigation hints
├── Content descriptions
└── Action announcements

Visual Accessibility:
├── High contrast mode support
├── Large text size adaptation
├── Color-blind friendly design
└── Reduced motion options

Interaction Accessibility:
├── Minimum tap target size (44pt)
├── Voice control compatibility
├── Switch control support
└── AssistiveTouch optimization
```

⚙️ 기술 구현 상세

6.1 iOS 앱 아키텍처

6.1.1 프로젝트 구조

```
ReadyRun/
├── App/
│   ├── AppDelegate.swift
│   ├── SceneDelegate.swift
│   └── Info.plist
├── Core/
│   ├── Networking/
│   │   ├── APIClient.swift
│   │   ├── APIEndpoints.swift
│   │   └── APIError.swift
│   ├── Database/
│   │   ├── CoreDataManager.swift
│   │   └── CacheManager.swift
│   ├── Utils/
│   │   ├── Extensions/
│   │   ├── Constants.swift
│   │   └── Helpers.swift
│   └── Security/
│       ├── Keychain.swift
│       └── AuthManager.swift
├── Features/
│   ├── Authentication/
│   ├── Home/
│   ├── Search/
│   ├── MarathonDetail/
│   ├── Favorites/
│   └── Profile/
├── Shared/
│   ├── Components/
│   ├── Models/
│   ├── ViewModels/
│   └── Services/
└── Resources/
    ├── Assets.xcassets
    ├── Localizable.strings
    └── GoogleService-Info.plist
```

6.1.2 아키텍처 패턴 (MVVM + Coordinator)

swift

```swift
// Coordinator Pattern for Navigation
protocol Coordinator {
    var navigationController: UINavigationController { get set }
    func start()
}

// MVVM Pattern
class MarathonListViewModel: ObservableObject {
    @Published var marathons: [Marathon] = []
    @Published var isLoading = false
    @Published var error: APIError?

    private let apiService: APIServiceProtocol
    private let locationService: LocationServiceProtocol

    func fetchMarathons() {
        // Implementation
    }
}

// Repository Pattern for Data Access
protocol MarathonRepositoryProtocol {
    func fetchMarathons(filters: MarathonFilters) async throws -> [Marathon]
    func getMarathonDetail(id: String) async throws -> MarathonDetail
    func addToFavorites(marathonId: String) async throws
}
```

6.1.3 상태 관리 (Combine + SwiftUI)

```swift
// Observable State Management
class AppState: ObservableObject {
    @Published var user: User?
    @Published var favorites: [String] = []
    @Published var searchFilters: SearchFilters = SearchFilters()
    @Published var locationPermission: LocationPermission = .notDetermined

    private var cancellables = Set<AnyCancellable>()

    init() {
        setupBindings()
    }

    private func setupBindings() {
        // Reactive state updates
    }
}
```

6.2 백엔드 아키텍처 (Node.js + Express)

6.2.1 프로젝트 구조

```
readyrun-backend/
├── src/
│   ├── controllers/
│   │   ├── marathonController.js
│   │   ├── userController.js
│   │   ├── favoriteController.js
│   │   └── adminController.js
│   ├── middleware/
│   │   ├── auth.js
│   │   ├── validation.js
│   │   ├── rateLimit.js
│   │   └── errorHandler.js
│   ├── routes/
│   │   ├── api/
│   │   │   ├── v1/
│   │   │   │   ├── marathons.js
│   │   │   │   ├── users.js
│   │   │   │   ├── favorites.js
│   │   │   │   └── admin.js
│   │   │   └── index.js
│   │   └── index.js
│   ├── services/
│   │   ├── marathonService.js
│   │   ├── userService.js
│   │   ├── notificationService.js
│   │   ├── crawlerService.js
│   │   └── locationService.js
│   ├── models/
│   │   ├── Marathon.js
│   │   ├── User.js
│   │   ├── Favorite.js
│   │   └── AdminUser.js
│   ├── utils/
│   │   ├── database.js
│   │   ├── logger.js
│   │   ├── helpers.js
│   │   └── constants.js
│   ├── config/
│   │   ├── database.js
│   │   ├── firebase.js
│   │   └── environment.js
│   └── app.js
├── scripts/
│   ├── migrate.js
│   ├── seed.js
│   └── crawler.js
│
```

```
├── tests/
│   ├── unit/
│   ├── integration/
│   └── e2e/
├── docs/
├── package.json
├── .env.example
└── vercel.json
```

6.2.2 API 구현 예시

javascript

```javascript
// Marathon Controller
const marathonController = {
  async getMarathons(req, res, next) {
    try {
      const filters = {
        page: parseInt(req.query.page) || 1,
        limit: Math.min(parseInt(req.query.limit) || 20, 100),
        country: req.query.country,
        distance: req.query.distance,
        dateFrom: req.query.date_from,
        dateTo: req.query.date_to,
        lat: parseFloat(req.query.lat),
        lng: parseFloat(req.query.lng),
        radius: parseInt(req.query.radius) || 50,
        tags: req.query.tags ? req.query.tags.split(',') : [],
        sort: req.query.sort || 'date_asc'
      };

      const result = await marathonService.getMarathons(filters, req.user?.id);

      res.json({
        success: true,
        data: {
          marathons: result.marathons,
          pagination: result.pagination
        }
      });
    } catch (error) {
      next(error);
    }
  },

  async getMarathonDetail(req, res, next) {
    try {
      const { id } = req.params;
      const marathon = await marathonService.getMarathonDetail(id, req.user?.id);

      if (!marathon) {
        return res.status(404).json({
          success: false,
          error: {
            code: 'MARATHON_NOT_FOUND',
            message: 'Marathon not found'
          }
        });
      }
```

```
        // Increment view count
        await marathonService.incrementViewCount(id);

        res.json({
          success: true,
          data: { marathon }
        });
      } catch (error) {
        next(error);
      }
    }
  };
```

6.2.3 데이터베이스 최적화

```sql
-- Performance Indexes
CREATE INDEX CONCURRENTLY idx_marathons_date_country_active
ON marathons (date_start, country)
WHERE status = 'active';

CREATE INDEX CONCURRENTLY idx_marathons_location_active
ON marathons USING GIST (
  ll_to_earth(latitude, longitude)
) WHERE status = 'active';

CREATE INDEX CONCURRENTLY idx_marathons_search_vector
ON marathons USING GIN (
  to_tsvector('english', name || ' ' || city || ' ' || country)
);

-- Materialized View for Popular Marathons
CREATE MATERIALIZED VIEW popular_marathons AS
SELECT
  m.*,
  f.favorite_count,
  m.view_count,
  (f.favorite_count * 0.7 + m.view_count * 0.3) AS popularity_score
FROM marathons m
LEFT JOIN (
  SELECT marathon_id, COUNT(*) as favorite_count
  FROM favorites
  GROUP BY marathon_id
) f ON m.id = f.marathon_id
WHERE m.status = 'active'
ORDER BY popularity_score DESC;

-- Refresh schedule (cron job)
REFRESH MATERIALIZED VIEW CONCURRENTLY popular_marathons;
```

6.3 AI 크롤러 시스템

6.3.1 크롤러 아키텍처

javascript

```javascript
// Crawler Service
class MarathonCrawler {
  constructor() {
    this.browser = null;
    this.sources = [
      'https://www.marathonguide.com',
      'https://www.runnersworld.com/races',
      'https://www.active.com/running',
      // Korean sources
      'https://www.marathon.pe.kr',
      'https://www.runday.co.kr'
    ];
  }

  async initialize() {
    this.browser = await puppeteer.launch({
      headless: true,
      args: ['--no-sandbox', '--disable-setuid-sandbox']
    });
  }

  async crawlSource(source) {
    const page = await this.browser.newPage();
    try {
      await page.goto(source, { waitUntil: 'networkidle2' });

      // Extract marathon data using selectors
      const marathons = await page.evaluate(() => {
        // Site-specific extraction logic
        return extractMarathonData();
      });

      return marathons.map(marathon => ({
        ...marathon,
        source: source,
        confidence_score: this.calculateConfidence(marathon),
        scraped_at: new Date()
      }));

    } catch (error) {
      console.error(`Crawling error for ${source}:`, error);
      return [];
    } finally {
      await page.close();
    }
  }
  calculateConfidence(marathon) {
    let score = 100;

    // Reduce score for missing critical fields
    if (!marathon.name) score -= 50;
    if (!marathon.date) score -= 30;
    if (!marathon.location) score -= 20;

    // Bonus for complete data
    if (marathon.registration_url) score += 10;
    if (marathon.official_website) score += 5;

    return Math.max(0, Math.min(100, score));
  }

  async processAndStore(marathons) {
    for (const marathon of marathons) {
      try {
        // Check for duplicates
        const existing = await this.findDuplicate(marathon);

        if (existing) {
          await this.updateExisting(existing.id, marathon);
        } else {
          await this.createNew(marathon);
        }
      } catch (error) {
        console.error('Error processing marathon:', error);
      }
    }
  }
}
```

6.3.2 데이터 검증 및 정규화

javascript

```javascript
// Data Validation Service
class DataValidator {
  static validateMarathon(data) {
    const errors = [];

    // Required fields
    if (!data.name || data.name.length < 3) {
      errors.push('Marathon name is required and must be at least 3 characters');
    }

    if (!data.date || !this.isValidDate(data.date)) {
      errors.push('Valid date is required');
    }

    if (!data.location || !data.location.country) {
      errors.push('Location with country is required');
    }

    // Distance validation
    if (!data.distances || !Array.isArray(data.distances)) {
      errors.push('At least one distance is required');
    }

    // Price validation
    if (data.registration_fee && (data.registration_fee < 0 || data.registration_fee >
      errors.push('Registration fee must be between 0 and 1,000,000');
    }

    return {
      isValid: errors.length === 0,
      errors
    };
  }

  static normalizeData(data) {
    return {
      name: this.normalizeString(data.name),
      slug: this.generateSlug(data.name),
      date_start: this.normalizeDate(data.date),
      country: this.normalizeCountry(data.location.country),
      city: this.normalizeString(data.location.city),
      distances: this.normalizeDistances(data.distances),
      registration_fee: this.normalizePrice(data.registration_fee),
      currency: this.normalizeCurrency(data.currency),
      // ... other fields
    };
  }

  static generateSlug(name) {
    return name
      .toLowerCase()
      .replace(/[^a-z0-9\s-]/g, '')
      .replace(/\s+/g, '-')
      .replace(/-+/g, '-')
      .trim('-');
  }
}
```

6.4 실시간 알림 시스템

6.4.1 Firebase Cloud Messaging 구현

javascript

```javascript
// Notification Service
class NotificationService {
  constructor() {
    this.fcm = admin.messaging();
  }

  async sendRegistrationDeadlineReminder(userId, marathon) {
    const user = await User.findById(userId);
    if (!user.push_notifications_enabled) return;

    const message = {
      token: user.fcm_token,
      notification: {
        title: '등록 마감 임박!',
        body: `${marathon.name} 등록이 3일 후 마감됩니다.`
      },
      data: {
        type: 'registration_deadline',
        marathon_id: marathon.id,
        deep_link: `readyrun://marathon/${marathon.id}`
      },
      apns: {
        payload: {
          aps: {
            sound: 'default',
            badge: 1,
            'mutable-content': 1
          }
        }
      }
    };

    try {
      await this.fcm.send(message);

      // Store notification in database
      await this.storeNotification({
        user_id: userId,
        type: 'registration_deadline',
        title: message.notification.title,
        body: message.notification.body,
        marathon_id: marathon.id,
        sent_at: new Date()
      });

    } catch (error) {
      console.error('FCM send error:', error);
    }
  }

  async scheduleNotifications() {
    // Find marathons with registration deadlines in 3 days
    const upcomingDeadlines = await Marathon.findUpcomingDeadlines(3);

    for (const marathon of upcomingDeadlines) {
      const interestedUsers = await this.getInterestedUsers(marathon.id);

      for (const user of interestedUsers) {
        await this.sendRegistrationDeadlineReminder(user.id, marathon);
      }
    }
  }
}
```

6.4.2 iOS 알림 처리

swift

```swift
// Push Notification Handler
class NotificationManager: NSObject, UNUserNotificationCenterDelegate {
    static let shared = NotificationManager()

    func requestPermission() {
        UNUserNotificationCenter.current().requestAuthorization(
            options: [.alert, .badge, .sound]
        ) { granted, error in
            if granted {
                DispatchQueue.main.async {
                    UIApplication.shared.registerForRemoteNotifications()
                }
            }
        }
    }

    // Handle notification when app is in foreground
    func userNotificationCenter(
        _ center: UNUserNotificationCenter,
        willPresent notification: UNNotification,
        withCompletionHandler completionHandler: @escaping (UNNotificationPresentation
    ) {
        completionHandler([.banner, .sound, .badge])
    }

    // Handle notification tap
    func userNotificationCenter(
        _ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: @escaping () -> Void
    ) {
        let userInfo = response.notification.request.content.userInfo

        if let marathonId = userInfo["marathon_id"] as? String,
           let deepLink = userInfo["deep_link"] as? String {
            handleDeepLink(deepLink)
        }

        completionHandler()
    }

    private func handleDeepLink(_ urlString: String) {
        guard let url = URL(string: urlString) else { return }

        // Parse and navigate to appropriate screen
        DeepLinkManager.shared.handle(url)
    }
}
```

💰 수익화 전략 (AdMob 구현)

7.1 광고 배치 전략

7.1.1 광고 유형별 배치

```
Banner Ads (320x50):
├── Home Screen: 하단 (스크롤 시 고정)
├── Search Results: 매 10개 결과마다
├── Marathon Detail: 중간 섹션 사이
└── Favorites: 리스트 하단

Interstitial Ads:
├── 앱 시작 시 (3회 사용 후)
├── 상세 화면 진입 시 (5회마다)
└── 검색 결과 페이지 전환 시

Native Ads:
├── Home Screen 추천 섹션에 통합
├── Search Results에 자연스럽게 삽입
└── Related Events 섹션

Rewarded Ads:
├── Premium 기능 체험 (필터 추가 옵션)
├── 광고 제거 일시적 혜택
└── 특별 콘텐츠 접근
```

7.1.2 AdMob 구현 코드

swift

```swift
// iOS AdMob Implementation
import GoogleMobileAds

class AdManager: ObservableObject {
    static let shared = AdManager()
    @Published var isAdLoaded = false

    private var bannerView: GADBannerView?
    private var interstitial: GADInterstitialAd?
    private var rewardedAd: GADRewardedAd?

    init() {
        GADMobileAds.sharedInstance().start(completionHandler: nil)
        loadAds()
    }

    func createBannerView() -> GADBannerView {
        let bannerView = GADBannerView(adSize: GADAdSizeBanner)
        bannerView.adUnitID = "ca-app-pub-XXXXXXXXXXXXXXXX/XXXXXXXXXX"
        bannerView.rootViewController = UIApplication.shared.windows.first?.rootViewCo
        bannerView.load(GADRequest())

        return bannerView
    }

    func loadInterstitial() {
        let request = GADRequest()
        GADInterstitial
```