# FitTrack: Simple Fitness Companion

Project Documentation and Self-Evaluation

## Product Definition Statement

FitTrack is a streamlined fitness tracking application designed for iOS that helps users monitor their daily nutrition and exercise activities. It differentiates itself through a simplified, user-friendly interface while maintaining powerful features like HealthKit integration and barcode scanning for food logging.

## Implemented Features

1. User Profile Management
    - Personal information storage (height, weight, age, gender)
    - Custom fitness goals setting
    - Macronutrient targets configuration
2. Food Logging System
    - Manual food entry with nutritional information
    - Barcode scanning integration with OpenFoodFacts API
    - Meal categorization (Breakfast, Lunch, Dinner, Snacks)
    - Daily calorie and macro tracking
3. Exercise Tracking
    - Manual exercise logging
    - Duration and intensity tracking
    - Calorie burn calculations
    - Notes for workout details
4. HealthKit Integration
    - Step counting
    - Active energy tracking
    - Resting energy monitoring
    - Background health data sync
5. Progress Monitoring
    - Weight tracking over time
    - Calorie intake visualization
    - Exercise progress tracking
    - Achievement tracking
6. Modern UI/UX
    - Tab-based navigation

- Interactive calendar for date selection
- Smooth animations and transitions
- Responsive design

7. Data Management
    - Firebase Realtime Database integration
    - Real-time data synchronization
    - Secure user authentication
8. Widget Support
    - Macro tracking widget
    - Daily progress visualization
    - Real-time updates

# Technical Architecture

**The app follows a clean MVVM architecture with clear separation of concerns:**

1. Core Managers
    - **AuthManager**: Handles user authentication and session management
    - **BiometricManager**: Manages biometric authentication (Face ID/Touch ID)
    - **CameraManager**: Handles camera access and permissions for barcode scanning functionality
    - **ExerciseLogManager**: Manages exercise tracking and storage
    - **FirebaseManager**: Manages Firebase configuration, authentication, and real-time database operations
    - **FoodLogManager**: Handles food logging operations and database interactions
    - **HealthDataManager**: Manages health-related data storage and synchronization with Firebase
    - **HealthKitManager**: Controls HealthKit integration and data sync
    - **NotificationManager**: Controls local notifications and reminders
    - **WeightLogManager**: Handles weight logging
    - **UserProfileManager**: Manages user profile data
2. Data Models
    - **UserProfile**: User information and preferences
    - **FoodLogItem**: Food entry data structure
    - **ExerciseLogItem**: Exercise entry data structure
    - **WeightLogItem**: Weight tracking data structure
    - **HealthDataItem**: Data structure for storing health metrics
    - **OpenFoodFactsResponse**: API response model for the OpenFoodFacts barcode scanning service
    - **Product**: Model representing a food product from OpenFoodFacts
    - **Nutriments**: Nutritional information model
    - **FoodScanResult**: Model for processed barcode scan results

- **MealSection**: Data structure for organizing food entries by meal type
- **CalendarDay**: Model for calendar date representation
- **FoodItem**: Model for individual food entries

## 3. View Structure

- Main Views:
    - SplashScreenView
    - iCloudSignInView
    - HomeView
    - DashboardPage
    - LogPage
    - ProgressPage
    - ProfilePage
    - OnboardingView
    - PermissionsSetupView
    - ProfileSetupView
    - TermsView
    - WelcomeView
    - GoalsSetupView
- Supporting Views:
    - BarcodeScannerView
    - ExerciseEntryView
    - ExerciseLogSectionView
    - FoodEntryView
    - HealthSectionView
    - MealSectionView
    - ScannedFoodEntryView
    - WeightEntryView
    - WeightLogSectionView
    - BiometricUnlockView
    - ViewExtensions
    - FeatureRow
    - TermsSection
    - ProfileFormSection
    - PermissionCardButtonStyle
    - PermissionCard
    - GoalField
    - StatCard
    - ActivitySummaryView
    - WeightProgressView
    - PersonalInfoEditor
    - MacroGoalsEditor
    - DateButton
    - DailySummaryView

4. Widget Implementation
    - **MacroWidget**: Displays macro tracking information
    - **WidgetDataManager**: Handles widget data updates

## Self-Evaluation and Grade Justification

### This project demonstrates A-level quality through:

1. Software Design Patterns & Architecture
    - Implemented MVVM architecture with clear separation of concerns
    - Used protocol-oriented programming and delegation patterns
    - Created reusable components with loose coupling
    - Utilized property wrappers and proper state management
2. iOS Technologies
    - Complex HealthKit integration with background sync
    - Custom camera interface for barcode scanning
    - Widget extension with real-time updates
    - SwiftUI implementation following latest best practices
3. Data Management
    - Firebase Realtime Database with proper security rules
    - Efficient data structure design
    - Real-time synchronization
    - Proper error handling
4. User Experience
    - Intuitive interface following iOS design guidelines
    - Smooth animations and transitions
    - Proper loading states and error handling
    - Accessibility considerations
    - Biometric authentication for security

## Above and Beyond Features

### The project exceeds basic requirements through:

1. Advanced Technical Implementation
    - Real-time data synchronization with Firebase
    - Complex HealthKit integration with background updates
    - Custom camera interface for barcode scanning
    - Widget extension with live data updates
2. Enhanced User Experience
    - Biometric authentication for security

- Offline functionality
- Custom animations and transitions
- Comprehensive error handling

### 3. Additional Features

- Barcode scanning integration with OpenFoodFacts API
- Interactive data visualizations
- Widget support for quick data access
- Custom calendar implementation

## Third-Party Frameworks and Dependencies

### 1. Firebase SDK

- Purpose: Backend services and real-time database
- Source: Google Firebase
- Documentation: https://firebase.google.com/docs/ios/setup

### 2. HealthKit

- Purpose: Health and fitness data integration
- Source: Apple
- Documentation: https://developer.apple.com/documentation/healthkit

### 3. CloudKit

- Purpose: iCloud integration and data sync
- Source: Apple
- Documentation: https://developer.apple.com/icloud/cloudkit/

### 4. WidgetKit

- Purpose: iOS home screen widget implementation
- Source: Apple
- Documentation: https://developer.apple.com/documentation/widgetkit

### 5. UserNotifications

- Purpose: Local notification handling and scheduling
- Source: Apple
- Documentation: https://developer.apple.com/documentation/usernotifications

### 6. OpenFoodFacts API

- Purpose: Food database and barcode scanning
- Source: Open Food Facts
- Documentation: https://world.openfoodfacts.org/data

## Future Improvements

### UI/UX Improvements

- Enhanced calendar interaction with centered selection
- Haptic feedback implementation
- Improved month navigation
- Refined visual design

### Feature Additions

- Custom QR code generation for frequent meals
- Advanced progress analytics
- Expanded exercise tracking options
- Enhanced widget customization

### Performance Optimizations

- Improved data caching
- Better offline functionality
- Reduced network requests
- Enhanced battery efficiency

## Conclusion

The FitTrack app demonstrates A-level work through its comprehensive feature set, robust implementation of iOS technologies, and attention to user experience. While there are some known issues to address, the core functionality is solid and provides a strong foundation for future improvements.