

iOS Entegrasyonu Teknik Dokümantasyonu

Bu doküman, Codeyzer Ekstre Takip uygulamasının Android'den iOS'a taşınması için gerekli tüm teknik detayları içerir.

1. Mevcut Proje Yapısı

1.1 Capacitor Yapılandırması

Versiyon: Capacitor 7.x

```
// capacitor.config.ts (mevcut)
const config: CapacitorConfig = {
  appId: 'com.codeyzer.ekstre',
  appName: 'Codeyzer Ekstre Takip',
  webDir: 'dist'
};
```

iOS için gerekli güncellemeler:

```
const config: CapacitorConfig = {
  appId: 'com.codeyzer.ekstre',
  appName: 'Codeyzer Ekstre Takip',
  webDir: 'dist',
  ios: {
    contentInset: 'automatic',
    preferredContentMode: 'mobile',
    backgroundColor: '#ffffff'
  }
};
```

1.2 Mevcut Bağımlılıklar

Kategori	Paket	Versiyon
Capacitor Core	@capacitor/core	^7.2.0
Capacitor CLI	@capacitor/cli	^7.2.0
Capacitor Android	@capacitor/android	^7.2.0
Capacitor iOS	@capacitor/ios	Eklenmeli
UI Framework	@ionic/react	^8.5.5
State Management	redux-toolkit	^2.7.0

Kategori	Paket	Versiyon
Firebase	firebase	^11.6.1

2. Native Plugin Envanteri

Uygulama 5 özel Capacitor plugin'i kullanmaktadır. Her biri Android'de Java ile yazılmış olup iOS için Swift'e çevrilmesi gerekmektedir.

2.1 Plugin Özeti

Plugin	Android Dosya	iOS Karşılığı	Durum
GoogleAuthPlugin	GoogleAuthPlugin.java	GoogleAuthPlugin.swift	Yazılacak
SmsReaderPlugin	SmsReaderPlugin.java	Alternatif Gerekli	SMS iOS'ta yasaklı
PdfParserPlugin	PdfParserPlugin.java	PdfParserPlugin.swift	Yazılacak
OcrPlugin	OcrPlugin.java	OcrPlugin.swift	Yazılacak
SecureStoragePlugin	SecureStoragePlugin.java	SecureStoragePlugin.swift	Yazılacak

3. Plugin Detaylı Analizi ve iOS Dönüşümü

3.1 GoogleAuthPlugin

Android Implementasyonu (Mevcut)

Dosya: android/app/src/main/java/com/codeyzer/ekstre/GoogleAuthPlugin.java

Kullanılan Android Kütüphaneler:

- com.google.android.gms.auth.api.signin - Google Sign-In
- com.google.firebase.auth - Firebase Authentication
- com.google.api.services.gmail - Gmail API
- com.google.api.services.calendar - Calendar API

Plugin Metodları:

```
@PluginMethod
public void signIn(PluginCall call);

@PluginMethod
public void trySilentSignIn(PluginCall call);

@PluginMethod
public void signOut(PluginCall call);
```

```

@PluginMethod
public void createCalendarEvent(PluginCall call);

@PluginMethod
public void searchCalendarEvents(PluginCall call);

@PluginMethod
public void searchGmailMessages(PluginCall call);

@PluginMethod
public void getGmailMessageDetails(PluginCall call);

@PluginMethod
public void getGmailAttachment(PluginCall call);

```

OAuth Yapılandırması:

```

WEB_CLIENT_ID = "1008857567754-
2s7hevrbudal3m8qju85g31souc8v4g5.apps.googleusercontent.com"

```

Scopes:

- <https://www.googleapis.com/auth/gmail.readonly>
- <https://www.googleapis.com/auth/calendar.events>

iOS Swift Implementasyonu (Yazılacak)

Gerekli CocoaPods:

```

pod 'GoogleSignIn', '~> 7.0'
pod 'FirebaseAuth', '~> 10.0'
pod 'GoogleAPIClientForRESTCore', '~> 3.0'
pod 'GoogleAPIClientForREST/Gmail', '~> 3.0'
pod 'GoogleAPIClientForREST/Calendar', '~> 3.0'

```

Swift Plugin Yapısı:

```

// ios/App/App/Plugins/GoogleAuth/GoogleAuthPlugin.swift

import Capacitor
import GoogleSignIn
import FirebaseAuth
import GoogleAPIClientForREST

@objc(GoogleAuthPlugin)
public class GoogleAuthPlugin: CAPPlugin {

    private var signInConfig: GIDConfiguration?

```

```
private var currentUser: GIDGoogleUser?

public override func load() {
    // Google Sign-In yapılandırması
    guard let clientID = Bundle.main.object(forKey:
"GIDClientID") as? String else {
        return
    }
    signInConfig = GIDConfiguration(clientID: clientID)
}

@objc func signIn(_ call: CAPPluginCall) {
    DispatchQueue.main.async {
        guard let config = self.signInConfig,
            let presentingVC = self.bridge?.viewController else {
            call.reject("Configuration error")
            return
        }

        let scopes = [
            "https://www.googleapis.com/auth/gmail.readonly",
            "https://www.googleapis.com/auth/calendar.events"
        ]

        GIDSignIn.sharedInstance.signIn(
            with: config,
            presenting: presentingVC,
            hint: nil,
            additionalScopes: scopes
        ) { user, error in
            if let error = error {
                call.reject(error.localizedDescription)
                return
            }

            guard let user = user else {
                call.reject("No user returned")
                return
            }

            self.currentUser = user

            // Firebase ile authenticate
            guard let idToken = user.idToken?.tokenString else {
                call.reject("No ID token")
                return
            }

            let credential = GoogleAuthProvider.credential(
                withIDToken: idToken,
                accessToken: user.accessToken.tokenString
            )

            Auth.auth().signIn(with: credential) { authResult, error in
```

```

        if let error = error {
            call.reject(error.localizedDescription)
            return
        }

        call.resolve([
            "id": user.userID ?? "",
            "name": user.profile?.name ?? "",
            "email": user.profile?.email ?? "",
            "imageUrl": user.profile?.imageUrl(withDimension:
100)?.absoluteString ?? "",
            "idToken": idToken,
            "accessToken": user.accessToken.tokenString
        ])
    }
}

@objc func trySilentSignIn(_ call: CAPPluginCall) {
    GIDSignIn.sharedInstance.restorePreviousSignIn { user, error in
        if let error = error {
            call.reject(error.localizedDescription,
"SILENT_SIGN_IN_FAILED")
            return
        }

        guard let user = user else {
            call.reject("No previous sign-in", "NO_PREVIOUS_SIGN_IN")
            return
        }

        self.currentUser = user

        call.resolve([
            "id": user.userID ?? "",
            "name": user.profile?.name ?? "",
            "email": user.profile?.email ?? "",
            "imageUrl": user.profile?.imageUrl(withDimension:
100)?.absoluteString ?? "",
            "idToken": user.idToken?.tokenString ?? "",
            "accessToken": user.accessToken.tokenString
        ])
    }
}

@objc func signOut(_ call: CAPPluginCall) {
    GIDSignIn.sharedInstance.signOut()
    try? Auth.auth().signOut()
    currentUser = nil
    call.resolve()
}

```

// Gmail ve Calendar metodları ayrı handler'lara delege edilecek

```

@objc func searchGmailMessages(_ call: CAPPluginCall) {
    // GmailHandler.swift'e delege et
}

@objc func getGmailMessageDetails(_ call: CAPPluginCall) {
    // GmailHandler.swift'e delege et
}

@objc func getGmailAttachment(_ call: CAPPluginCall) {
    // GmailHandler.swift'e delege et
}

@objc func createCalendarEvent(_ call: CAPPluginCall) {
    // CalendarHandler.swift'e delege et
}

@objc func searchCalendarEvents(_ call: CAPPluginCall) {
    // CalendarHandler.swift'e delege et
}
}

```

Gmail Handler (Swift):

```

// ios/App/App/Plugins/GoogleAuth/GmailHandler.swift

import GoogleAPIClientForREST

class GmailHandler {
    private let gmailService = GTLRGmailService()

    init(user: GIDGoogleUser) {
        gmailService.authorizer = user.fetcherAuthorizer
    }

    func searchMessages(query: String, completion: @escaping
(Result<[String: Any], Error>) -> Void) {
        let listQuery = GTLRGmailQuery_UsersMessagesList.query(withUserId:
"me")
        listQuery.q = query

        gmailService.executeQuery(listQuery) { ticket, response, error in
            if let error = error {
                completion(.failure(error))
                return
            }

            guard let messagesResponse = response as?
GTLRGmail_ListMessagesResponse else {
                completion(.failure(NSError(domain: "GmailHandler", code:
-1)))
                return
            }
        }
    }
}

```

```

        var messages: [[String: Any]] = []
        for message in messagesResponse.messages ?? [] {
            messages.append([
                "id": message.identifier ?? "",
                "threadId": message.threadId ?? ""
            ])
        }

        completion(.success([
            "messages": messages,
            "nextPageToken": messagesResponse.nextPageToken ?? "",
            "resultSizeEstimate": messagesResponse.resultSizeEstimate
        ]))
    }
}

func getMessageDetails(messageId: String, completion: @escaping
(Result<[String: Any], Error>) -> Void) {
    let query = GTLRGmailQuery_UsersMessagesGet.query(withUserId: "me",
identifier: messageId)
    query.format = "full"

    gmailService.executeQuery(query) { ticket, response, error in
        if let error = error {
            completion(.failure(error))
            return
        }

        guard let message = response as? GTLRGmail_Message else {
            completion(.failure(NSError(domain: "GmailHandler", code:
-1)))
            return
        }

        // GTLRGmail_Message'ı dictionary'ye dönüştür
        completion(.success(self.messageToDict(message)))
    }
}

func getAttachment(messageId: String, attachmentId: String, completion:
@escaping (Result<[String: Any], Error>) -> Void) {
    let query = GTLRGmailQuery_UsersMessagesAttachmentsGet.query(
        withUserId: "me",
        messageId: messageId,
        identifier: attachmentId
    )

    gmailService.executeQuery(query) { ticket, response, error in
        if let error = error {
            completion(.failure(error))
            return
        }
    }
}

```

```

        guard let attachment = response as? GTLRGmail_MessagePartBody
    else {
        completion(.failure(NSError(domain: "GmailHandler", code:
-1)))
        return
    }

    completion(.success([
        "attachmentId": attachment.attachmentId ?? "",
        "size": attachment.size ?? 0,
        "data": attachment.data ?? "" // Base64url encoded
    ]))
    }
}

private func messageToDict(_ message: GTLRGmail_Message) -> [String:
Any] {
    // Message objesini dictionary'ye dönüştür
    return [
        "id": message.identifier ?? "",
        "threadId": message.threadId ?? "",
        "labelIds": message.labelIds ?? [],
        "snippet": message.snippet ?? "",
        "internalDate": message.internalDate ?? "",
        "payload": payloadToDict(message.payload),
        "sizeEstimate": message.sizeEstimate ?? 0
    ]
}

private func payloadToDict(_ payload: GTLRGmail_MessagePart?) ->
[String: Any]? {
    guard let payload = payload else { return nil }

    var headers: [[String: String]] = []
    for header in payload.headers ?? [] {
        headers.append([
            "name": header.name ?? "",
            "value": header.value ?? ""
        ])
    }

    return [
        "partId": payload.partId ?? "",
        "mimeType": payload.mimeType ?? "",
        "filename": payload.filename ?? "",
        "headers": headers,
        "body": [
            "attachmentId": payload.body?.attachmentId ?? "",
            "size": payload.body?.size ?? 0,
            "data": payload.body?.data ?? ""
        ],
        "parts": payload.parts?.map { partToDict($0) } ?? []
    ]
}

```

```
}  
}
```

Calendar Handler (Swift):

```
// ios/App/App/Plugins/GoogleAuth/CalendarHandler.swift  
  
import GoogleAPIClientForREST  
  
class CalendarHandler {  
    private let calendarService = GTLRCalendarService()  
  
    init(user: GIDGoogleUser) {  
        calendarService.authorizer = user.fetcherAuthorizer  
    }  
  
    func createEvent(  
        summary: String,  
        description: String,  
        startTimeIso: String,  
        endTimeIso: String,  
        timeZone: String = "Europe/Istanbul",  
        completion: @escaping (Result<[String: Any], Error>) -> Void  
    ) {  
        let event = GTLRCalendar_Event()  
        event.summary = summary  
        event.descriptionProperty = description  
  
        let dateFormatter = ISO8601DateFormatter()  
  
        let startDateTime = GTLRCalendar_EventDateTime()  
        startDateTime.dateTime = GTLRDateTime(date: dateFormatter.date(from: startTimeIso)!)  
        startDateTime.timeZone = timeZone  
        event.start = startDateTime  
  
        let endDateTime = GTLRCalendar_EventDateTime()  
        endDateTime.dateTime = GTLRDateTime(date: dateFormatter.date(from: endTimeIso)!)  
        endDateTime.timeZone = timeZone  
        event.end = endDateTime  
  
        // Hatırlatıcı ekle  
        let reminder = GTLRCalendar_EventReminder()  
        reminder.method = "popup"  
        reminder.minutes = 60 * 24 // 1 gün önce  
  
        let reminders = GTLRCalendar_Event_Reminders()  
        reminders.useDefault = false  
        reminders.overrides = [reminder]  
        event.reminders = reminders  
    }  
}
```

```

        let query = GTLRCalendarQuery_EventsInsert.query(withObject: event,
calendarId: "primary")

        calendarService.executeQuery(query) { ticket, response, error in
            if let error = error {
                completion(.failure(error))
                return
            }

            guard let createdEvent = response as? GTLRCalendar_Event else {
                completion(.failure(NSError(domain: "CalendarHandler",
code: -1)))
                return
            }

            completion(.success([
                "id": createdEvent.identifier ?? "",
                "htmlLink": createdEvent.htmlLink ?? "",
                "summary": createdEvent.summary ?? ""
            ]))
        }
    }

    func searchEvents(appId: String, completion: @escaping (Result<[String:
Any], Error>) -> Void) {
        // AppID'den tarihi çıkar (format: YYYY-MM-DD-...)
        let pattern = #"(\d{4}-\d{2}-\d{2})"#
        guard let regex = try? NSRegularExpression(pattern: pattern),
            let match = regex.firstMatch(in: appId, range:
NSRange(appId.startIndex..., in: appId)),
            let range = Range(match.range(at: 1), in: appId) else {
            completion(.success(["eventFound": false]))
            return
        }

        let dateString = String(appId[range])
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "yyyy-MM-dd"

        guard let date = dateFormatter.date(from: dateString) else {
            completion(.success(["eventFound": false]))
            return
        }

        // Aynı günün başı ve sonu
        let calendar = Calendar.current
        let startOfDay = calendar.startOfDay(for: date)
        let endOfDay = calendar.date(byAdding: .day, value: 1, to:
startOfDay)!

        let query = GTLRCalendarQuery_EventsList.query(withCalendarId:
"primary")
        query.timeMin = GTLRDateTime(date: startOfDay)
        query.timeMax = GTLRDateTime(date: endOfDay)
    }

```

```

        query.singleEvents = true

        calendarService.executeQuery(query) { ticket, response, error in
            if let error = error {
                completion(.failure(error))
                return
            }

            guard let events = response as? GTLRCalendar_Events else {
                completion(.success(["eventFound": false]))
                return
            }

            // Description'da appId'yi ara
            let eventFound = events.items?.contains { event in
                event.descriptionProperty?.contains(appId) ?? false
            } ?? false

            completion(.success(["eventFound": eventFound]))
        }
    }
}

```

3.2 SmsReaderPlugin

UYARI: iOS'ta SMS okuma API'si bulunmamaktadır. Apple, güvenlik ve gizlilik nedeniyle uygulamaların SMS içeriğine erişmesine izin vermemektedir.

Android Implementasyonu (Mevcut)

Dosya: `android/app/src/main/java/com/codeyzer/ekstre/SmsReaderPlugin.java`

```

@CapacitorPlugin(
    name = "SmsReader",
    permissions = {
        @Permission(
            alias = "readSms",
            strings = { Manifest.permission.READ_SMS }
        )
    }
)
public class SmsReaderPlugin extends Plugin {

    @PluginMethod
    public void getMessages(PluginCall call) {
        // ContentResolver ile SMS Inbox'tan mesaj okuma
        // GLOB pattern ile filtreleme
    }

    @PluginMethod

```

```
public void checkPermissions(PluginCall call);

@PluginMethod
public void requestPermissions(PluginCall call);
}
```

iOS Alternatif Stratejileri

Seenek 1: Sadece Gmail ile Devam Et

- SMS yerine tüm banka bildirimlerini Gmail üzerinden al
- Kullanıcıları banka e-posta bildirimlerini etkinleştirmeye yönlendir
- En kolay ve hızlı çözüm

Seenek 2: Push Notification Service Extension

- Banka uygulamalarından gelen push bildirimleri yakala
- `UNNotificationServiceExtension` kullan
- Sınırlı ve banka uygulamasına bağımlı

Seenek 3: Clipboard Monitoring (Kullanıcı Müdahalesi)

- Kullanıcı SMS'i kopyalasın
- Uygulama clipboard'ı okusun
- Kötü kullanıcı deneyimi

Önerilen Yaklaşım:

```
// ios/App/App/Plugins/SmsReader/SmsReaderPlugin.swift

import Capacitor

@objc(SmsReaderPlugin)
public class SmsReaderPlugin: CAPPlugin {

    @objc func checkPermissions(_ call: CAPPluginCall) {
        // iOS'ta SMS izni yok
        call.resolve([
            "readSms": "denied",
            "platform": "ios",
            "message": "SMS reading is not available on iOS. Please use
email notifications."
        ])
    }

    @objc func requestPermissions(_ call: CAPPluginCall) {
        call.reject(
            "SMS reading is not available on iOS",
            "PLATFORM_NOT_SUPPORTED",
            nil,
            [

```

```

        "platform": "ios",
        "suggestion": "Use Gmail integration for bank
notifications"
    ]
  )
}

@objc func getMessages(_ call: CAPPluginCall) {
  call.reject(
    "SMS reading is not available on iOS",
    "PLATFORM_NOT_SUPPORTED",
    nil,
    [
      "platform": "ios",
      "suggestion": "Use Gmail integration for bank
notifications"
    ]
  )
}
}

```

TypeScript Tarafında Platform Kontrolü:

```

// src/services/sms-parsing/sms-processor.ts

import { Capacitor } from '@capacitor/core';

export class StatementProcessor {
  async fetchAndParseStatements(): Promise<ParsedStatement[]> {
    const platform = Capacitor.getPlatform();

    if (platform === 'ios') {
      // iOS'ta sadece email parsing
      console.log('iOS detected - using email only');
      return this.fetchEmailStatements();
    }

    // Android'de SMS + Email
    const smsStatements = await this.fetchSmsStatements();
    const emailStatements = await this.fetchEmailStatements();

    return [...smsStatements, ...emailStatements];
  }
}

```

3.3 PdfParserPlugin

Android Implementasyonu (Mevcut)

Dosya: android/app/src/main/java/com/codeyzer/ekstre/PdfParserPlugin.java

```
// Kullanılan kütüphane: com.tom-roush:pdfbox-android:2.0.27.0
@PluginMethod
public void parsePdfText(PluginCall call) {
    String base64Data = call.getString("base64Data");
    byte[] pdfBytes = Base64.decode(base64Data, Base64.DEFAULT);
    PDDocument document = PDDocument.load(pdfBytes);
    PDFTextStripper stripper = new PDFTextStripper();
    String text = stripper.getText(document);
    // ...
}
```

iOS Swift Implementasyonu

Yöntem 1: PDFKit (Native - Önerilen)

```
// ios/App/App/Plugins/PdfParser/PdfParserPlugin.swift

import Capacitor
import PDFKit

@objc(PdfParserPlugin)
public class PdfParserPlugin: CAPPlugin {

    @objc func parsePdfText(_ call: CAPPluginCall) {
        guard let base64Data = call.getString("base64Data") else {
            call.reject("Missing base64Data parameter")
            return
        }

        // Base64 decode
        guard let pdfData = Data(base64Encoded: base64Data) else {
            call.reject("Invalid base64 data")
            return
        }

        // PDFDocument oluştur
        guard let pdfDocument = PDFDocument(data: pdfData) else {
            call.resolve(["error": "Failed to load PDF document"])
            return
        }

        // Şifreli PDF kontrolü
        if pdfDocument.isEncrypted && !pdfDocument.isLocked {
            call.resolve(["error": "PDF is encrypted and cannot be processed"])
            return
        }
    }
}
```

```
// Tüm sayfalardan metin çıkar
var fullText = ""
for pageIndex in 0..
```

Yöntem 2: Vision Framework ile OCR (Taranmış PDF'ler için)

```
// Taranmış (görüntü tabanlı) PDF'ler için
import Vision

extension PdfParserPlugin {

    func extractTextWithOCR(from pdfDocument: PDFDocument, completion:
@escaping (String) -> Void) {
        var extractedText = ""
        let dispatchGroup = DispatchGroup()

        for pageIndex in 0..
```

```

        let request = VNRecognizeTextRequest { request, error in
            defer { dispatchGroup.leave() }

            guard let observations = request.results as?
[VNRecognizedTextObservation] else {
                return
            }

            for observation in observations {
                if let topCandidate =
observation.topCandidates(1).first {
                    extractedText += topCandidate.string + "\n"
                }
            }
        }

        request.recognitionLevel = .accurate
        request.recognitionLanguages = ["tr-TR", "en-US"]

        let handler = VNImageRequestHandler(cgImage: cgImage, options:
[:])
        try? handler.perform([request])
    }

    dispatchGroup.notify(queue: .main) {
        completion(extractedText)
    }
}
}

```

3.4 OcrPlugin

Android Implementasyonu (Mevcut)

Dosya: android/app/src/main/java/com/codezyer/ekstre/OcrPlugin.java

```

// Kullanılan: com.google.android.gms:play-services-mlkit-text-recognition
@PluginMethod
public void recognizeText(PluginCall call) {
    String imageSource = call.getString("imageSource");
    String sourceType = call.getString("sourceType", "path");

    InputImage image;
    if ("base64".equals(sourceType)) {
        byte[] imageBytes = Base64.decode(imageSource, Base64.DEFAULT);
        Bitmap bitmap = BitmapFactory.decodeByteArray(imageBytes, 0,
imageBytes.length);
        image = InputImage.fromBitmap(bitmap, 0);
    } else {

```

```
        // File path
        image = InputImage.fromFilePath(context, Uri.parse(imageSource));
    }

    TextRecognizer recognizer =
    TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS);
    recognizer.process(image)...
}
```

iOS Swift Implementasyonu

Yöntem 1: Vision Framework (Native - Önerilen)

```
// ios/App/App/Plugins/Ocr/OcrPlugin.swift

import Capacitor
import Vision
import UIKit

@objc(OcrPlugin)
public class OcrPlugin: CAPPlugin {

    @objc func recognizeText(_ call: CAPPluginCall) {
        guard let imageSource = call.getString("imageSource") else {
            call.reject("Missing imageSource parameter")
            return
        }

        let sourceType = call.getString("sourceType") ?? "path"

        // Görüntüyü yükle
        loadImage(from: imageSource, sourceType: sourceType) { [weak self]
result in
            switch result {
            case .success(let image):
                self?.performOCR(on: image, call: call)
            case .failure(let error):
                call.reject("Failed to load image: \(
(error.localizedDescription)")
            }
        }
    }

    private func loadImage(from source: String, sourceType: String,
completion: @escaping (Result<UIImage, Error>) -> Void) {
        DispatchQueue.global(qos: .userInitiated).async {
            if sourceType == "base64" {
                // Base64'ten yükle
                guard let imageData = Data(base64Encoded: source),
                    let image = UIImage(data: imageData) else {
                    completion(.failure(NSError(domain: "OcrPlugin", code:

```

```

-1, userInfo: [NSLocalizedStringKey: "Invalid base64 image data"])))
        return
    }
    completion(.success(image))
} else {
    // File path'ten yükle
    if source.hasPrefix("content://") ||
source.hasPrefix("file://") {
        guard let url = URL(string: source),
            let imageData = try? Data(contentsOf: url),
            let image = UIImage(data: imageData) else {
            completion(.failure(NSError(domain: "OcrPlugin",
code: -2, userInfo: [NSLocalizedStringKey: "Failed to load image from
URL"])))

            return
        }
        completion(.success(image))
    } else {
        // Local file path
        guard let image = UIImage(contentsOfFile: source) else
{
            completion(.failure(NSError(domain: "OcrPlugin",
code: -3, userInfo: [NSLocalizedStringKey: "Failed to load image from
file path"])))

            return
        }
        completion(.success(image))
    }
}
}
}

private func performOCR(on image: UIImage, call: CAPPluginCall) {
    guard let cgImage = image.cgImage else {
        call.reject("Failed to convert image")
        return
    }

    let request = VNRecognizeTextRequest { [weak self] request, error
in
        self?.handleOCRResult(request: request, error: error, call:
call)
    }

    // OCR ayarları
    request.recognitionLevel = .accurate
    request.recognitionLanguages = ["tr-TR", "en-US"]
    request.usesLanguageCorrection = true

    let handler = VNImageRequestHandler(cgImage: cgImage, options: [:])

    DispatchQueue.global(qos: .userInitiated).async {
        do {
            try handler.perform([request])

```

```

        } catch {
            DispatchQueue.main.async {
                call.reject("OCR failed: \
(error.localizedDescription)")
            }
        }
    }
}

private func handleOCRResult(request: VNRequest, error: Error?, call:
CAPPluginCall) {
    DispatchQueue.main.async {
        if let error = error {
            call.resolve([
                "success": false,
                "text": "",
                "error": error.localizedDescription
            ])
            return
        }

        guard let observations = request.results as?
[VNRecognizedTextObservation] else {
            call.resolve([
                "success": false,
                "text": "",
                "error": "No text observations found"
            ])
            return
        }

        var fullText = ""
        var blocks: [[String: Any]] = []

        for observation in observations {
            guard let topCandidate = observation.topCandidates(1).first
            else { continue }

            fullText += topCandidate.string + "\n"

            // Bounding box bilgisi
            let boundingBox = observation.boundingBox
            blocks.append([
                "text": topCandidate.string,
                "confidence": topCandidate.confidence,
                "boundingBox": [
                    "left": boundingBox.origin.x,
                    "top": 1 - boundingBox.origin.y -
boundingBox.height, // Koordinat dönüşümü
                    "width": boundingBox.width,
                    "height": boundingBox.height
                ]
            ])
        }
    }
}

```

```

        call.resolve([
            "success": true,
            "text": fullText.trimmingCharacters(in:
.whitespacesAndNewlines),
            "metadata": [
                "blocks": blocks,
                "blockCount": blocks.count
            ]
        ])
    }
}
}

```

Yöntem 2: Google ML Kit iOS SDK (Opsiyonel)

```

# Podfile
pod 'GoogleMLKit/TextRecognition', '~> 4.0'

```

```

import MLKitTextRecognition
import MLKitVision

// ML Kit ile implementasyon (Google consistency için)
func recognizeWithMLKit(image: UIImage, completion: @escaping
(Result<String, Error>) -> Void) {
    let visionImage = VisionImage(image: image)
    visionImage.orientation = image.imageOrientation

    let textRecognizer = TextRecognizer.textRecognizer()

    textRecognizer.process(visionImage) { result, error in
        if let error = error {
            completion(.failure(error))
            return
        }

        guard let result = result else {
            completion(.failure(NSError(domain: "MLKit", code: -1)))
            return
        }

        completion(.success(result.text))
    }
}

```

3.5 SecureStoragePlugin

Android Implementasyonu (Mevcut)

Dosya: `android/app/src/main/java/com/codeyzer/ekstre/SecureStoragePlugin.java`

```
// Android Keystore ile AES-256-GCM şifreleme
private static final String KEYSTORE_ALIAS = "codeyzer_ekstre_key";
private static final String ANDROID_KEYSTORE = "AndroidKeyStore";
private static final String AES_MODE = "AES/GCM/NoPadding";

@PluginMethod
public void encryptString(PluginCall call) {
    String data = call.getString("data");
    SecretKey key = getOrCreateKey();
    Cipher cipher = Cipher.getInstance(AES_MODE);
    cipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] iv = cipher.getIV();
    byte[] encrypted = cipher.doFinal(data.getBytes());
    // IV + encrypted data -> Base64
}

@PluginMethod
public void decryptString(PluginCall call) {
    // Base64 decode -> IV ayırıştır -> decrypt
}
```

iOS Swift Implementasyonu

```
// ios/App/App/Plugins/SecureStorage/SecureStoragePlugin.swift

import Capacitor
import Security
import CryptoKit

@objc(SecureStoragePlugin)
public class SecureStoragePlugin: CAPPlugin {

    private let keychainService = "com.codeyzer.ekstre.securestorage"
    private let keyAlias = "codeyzer_ekstre_key"

    @objc func encryptString(_ call: CAPPluginCall) {
        guard let data = call.getString("data") else {
            call.reject("Missing data parameter")
            return
        }

        do {
            let key = try getOrCreateKey()
            let encryptedData = try encrypt(data: data, with: key)
            call.resolve(["encryptedData": encryptedData])
        } catch {

```

```
        call.reject("Encryption failed: \(error.localizedDescription)")
    }
}

@objc func decryptString(_ call: CAPPluginCall) {
    guard let encryptedData = call.getString("encryptedData") else {
        call.reject("Missing encryptedData parameter")
        return
    }

    do {
        let key = try getOrCreateKey()
        let decryptedData = try decrypt(data: encryptedData, with: key)
        call.resolve(["decryptedData": decryptedData])
    } catch {
        call.reject("Decryption failed: \(error.localizedDescription)")
    }
}

// MARK: - Key Management

private func getOrCreateKey() throws -> SymmetricKey {
    // Keychain'den key'i al
    if let existingKey = retrieveKeyFromKeychain() {
        return existingKey
    }

    // Yeni key oluştur ve kaydet
    let newKey = SymmetricKey(size: .bits256)
    try storeKeyInKeychain(newKey)
    return newKey
}

private func retrieveKeyFromKeychain() -> SymmetricKey? {
    let query: [String: Any] = [
        kSecClass as String: kSecClassGenericPassword,
        kSecAttrService as String: keychainService,
        kSecAttrAccount as String: keyAlias,
        kSecReturnData as String: true,
        kSecMatchLimit as String: kSecMatchLimitOne
    ]

    var result: AnyObject?
    let status = SecItemCopyMatching(query as CFDictionary, &result)

    guard status == errSecSuccess,
        let keyData = result as? Data else {
        return nil
    }

    return SymmetricKey(data: keyData)
}

private func storeKeyInKeychain(_ key: SymmetricKey) throws {
```

```

        let keyData = key.withUnsafeBytes { Data($0) }

        let query: [String: Any] = [
            kSecClass as String: kSecClassGenericPassword,
            kSecAttrService as String: keychainService,
            kSecAttrAccount as String: keyAlias,
            kSecValueData as String: keyData,
            kSecAttrAccessible as String:
kSecAttrAccessibleWhenUnlockedThisDeviceOnly
        ]

        // Önce varsa sil
        SecItemDelete(query as CFDictionary)

        // Yeni key'i ekle
        let status = SecItemAdd(query as CFDictionary, nil)
        guard status == errSecSuccess else {
            throw NSError(domain: "SecureStorage", code: Int(status),
userInfo: [NSLocalizedDescriptionKey: "Failed to store key in Keychain"])
        }
    }

    // MARK: - Encryption/Decryption

    private func encrypt(data: String, with key: SymmetricKey) throws ->
String {
        guard let dataBytes = data.data(using: .utf8) else {
            throw NSError(domain: "SecureStorage", code: -1, userInfo:
[NSLocalizedDescriptionKey: "Invalid UTF-8 string"])
        }

        // AES-GCM şifreleme (CryptoKit)
        let sealedBox = try AES.GCM.seal(dataBytes, using: key)

        guard let combined = sealedBox.combined else {
            throw NSError(domain: "SecureStorage", code: -2, userInfo:
[NSLocalizedDescriptionKey: "Failed to combine sealed box"])
        }

        // Base64 encode (nonce + ciphertext + tag)
        return combined.base64EncodedString()
    }

    private func decrypt(data: String, with key: SymmetricKey) throws ->
String {
        guard let combined = Data(base64Encoded: data) else {
            throw NSError(domain: "SecureStorage", code: -3, userInfo:
[NSLocalizedDescriptionKey: "Invalid base64 data"])
        }

        // AES-GCM deşifreleme
        let sealedBox = try AES.GCM.SealedBox(combined: combined)
        let decryptedData = try AES.GCM.open(sealedBox, using: key)
    }

```

```
        guard let decryptedString = String(data: decryptedData, encoding:
.utf8) else {
            throw NSError(domain: "SecureStorage", code: -4, userInfo:
[NSLocalizedStringKey: "Failed to decode decrypted data"])
        }

        return decryptedString
    }
}
```

4. iOS Proje Kurulumu

4.1 Capacitor iOS Ekleme

```
# 1. iOS platform paketini ekle
npm install @capacitor/ios

# 2. iOS projesini oluştur
npx cap add ios

# 3. Senkronize et
npx cap sync ios
```

4.2 Proje Yapısı (Oluşturulacak)

```
ios/
├── App/
│   ├── App.xcodeproj/
│   ├── App.xcworkspace/
│   ├── Podfile
│   ├── Podfile.lock
│   └── App/
│       ├── AppDelegate.swift
│       ├── Info.plist
│       ├── Assets.xcassets/
│       └── Plugins/
│           ├── GoogleAuth/
│           │   ├── GoogleAuthPlugin.swift
│           │   ├── GoogleAuthPlugin.m
│           │   ├── GmailHandler.swift
│           │   └── CalendarHandler.swift
│           ├── SmsReader/
│           │   ├── SmsReaderPlugin.swift
│           │   └── SmsReaderPlugin.m
│           ├── PdfParser/
│           │   ├── PdfParserPlugin.swift
│           │   └── PdfParserPlugin.m
│           └── Ocr/
└── # Objective-C bridge
```

```

├── Tests/
├── Pods/
├── OcrPlugin.swift
├── OcrPlugin.m
├── SecureStorage/
├── SecureStoragePlugin.swift
├── SecureStoragePlugin.m

```

4.3 Podfile

```

# ios/App/Podfile

platform :ios, '14.0'
use_frameworks!

target 'App' do
  # Capacitor
  capacitor_pods

  # Google Sign-In & Firebase
  pod 'GoogleSignIn', '~> 7.0'
  pod 'FirebaseCore', '~> 10.0'
  pod 'FirebaseAuth', '~> 10.0'

  # Google API Client Libraries
  pod 'GoogleAPIClientForRESTCore', '~> 3.0'
  pod 'GoogleAPIClientForREST/Gmail', '~> 3.0'
  pod 'GoogleAPIClientForREST/Calendar', '~> 3.0'

  # Opsiyonel: Google ML Kit (OCR)
  # Vision framework yeterli, ancak Android ile tutarlılık istenirse:
  # pod 'GoogleMLKit/TextRecognition', '~> 4.0'
end

post_install do |installer|
  installer.pods_project.targets.each do |target|
    target.build_configurations.each do |config|
      config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = '14.0'
    end
  end
end

```

4.4 Info.plist Yapılandırması

```

<!-- ios/App/App/Info.plist -->

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">

```

```
<plist version="1.0">
<dict>
  <!-- Google Sign-In Client ID -->
  <key>GIDClientID</key>
  <string>YOUR_IOS_CLIENT_ID.apps.googleusercontent.com</string>

  <!-- URL Schemes -->
  <key>CFBundleURLTypes</key>
  <array>
    <!-- Google Sign-In URL Scheme -->
    <dict>
      <key>CFBundleURLSchemes</key>
      <array>

<string>com.googleusercontent.apps.YOUR_IOS_CLIENT_ID</string>
      </array>
      <key>CFBundleURLName</key>
      <string>Google Sign-In</string>
    </dict>
    <!-- App Custom URL Scheme (Deep Linking) -->
    <dict>
      <key>CFBundleURLSchemes</key>
      <array>
        <string>codeyzer-ekstre-takip</string>
      </array>
      <key>CFBundleURLName</key>
      <string>App Deep Link</string>
    </dict>
  </array>

  <!-- Privacy Descriptions -->
  <key>NSCameraUsageDescription</key>
  <string>Banka ekran görüntülerini taramak için kamera erişimi
gereklidir.</string>

  <key>NSPhotoLibraryUsageDescription</key>
  <string>Banka ekran görüntülerini seçmek için fotoğraf kitaplığı
erişimi gereklidir.</string>

  <key>NSPhotoLibraryAddUsageDescription</key>
  <string>Ekran görüntülerini kaydetmek için fotoğraf kitaplığı yazma
izni gereklidir.</string>

  <key>NSCalendarsUsageDescription</key>
  <string>Ödeme tarihlerini takviminize eklemek için takvim erişimi
gereklidir.</string>

  <!-- App Transport Security -->
  <key>NSAppTransportSecurity</key>
  <dict>
    <key>NSAllowsArbitraryLoads</key>
    <false/>
  </dict>
</dict>
</plist>
```

```
<!-- Backup Disabled (Security) -->
<key>UIFileSharingEnabled</key>
<false/>

<!-- Firebase Configuration File -->
<key>FirebaseAppDelegateProxyEnabled</key>
<false/>
</dict>
</plist>
```

4.5 AppDelegate.swift Güncellemeleri

```
// ios/App/App/AppDelegate.swift

import UIKit
import Capacitor
import GoogleSignIn
import FirebaseCore

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        // Firebase yapılandırması
        FirebaseApp.configure()

        return true
    }

    // Google Sign-In URL handling
    func application(_ app: UIApplication, open url: URL, options:
[UIApplication.OpenURLOptionsKey: Any] = [:]) -> Bool {

        // Google Sign-In
        if GIDSignIn.sharedInstance.handle(url) {
            return true
        }

        // Capacitor deep links
        return ApplicationDelegateProxy.shared.application(app, open: url,
options: options)
    }

    // Universal Links
    func application(_ application: UIApplication, continue userActivity:
NSUserActivity, restorationHandler: @escaping ([UIUserActivityRestoring]?)
```

```
-> Void) -> Bool {
    return AppDelegateProxy.shared.application(application,
    continue: userActivity, restorationHandler: restorationHandler)
}
}
```

4.6 Plugin Registrasyonu

```
// ios/App/App/Plugins/PluginRegistration.swift

import Capacitor

public func registerPlugins(on bridge: CAPBridgeProtocol) {
    bridge.registerPluginInstance(GoogleAuthPlugin())
    bridge.registerPluginInstance(SmsReaderPlugin())
    bridge.registerPluginInstance(PdfParserPlugin())
    bridge.registerPluginInstance(OcrPlugin())
    bridge.registerPluginInstance(SecureStoragePlugin())
}
```

Objective-C Bridge Dosyaları:

Her Swift plugin için bir `.m` bridge dosyası gereklidir:

```
// ios/App/App/Plugins/GoogleAuth/GoogleAuthPlugin.m

#import <Capacitor/Capacitor.h>

CAP_PLUGIN(GoogleAuthPlugin, "GoogleAuth",
    CAP_PLUGIN_METHOD(signIn, CAPPluginReturnPromise);
    CAP_PLUGIN_METHOD(trySilentSignIn, CAPPluginReturnPromise);
    CAP_PLUGIN_METHOD(signOut, CAPPluginReturnPromise);
    CAP_PLUGIN_METHOD(createCalendarEvent, CAPPluginReturnPromise);
    CAP_PLUGIN_METHOD(searchCalendarEvents, CAPPluginReturnPromise);
    CAP_PLUGIN_METHOD(searchGmailMessages, CAPPluginReturnPromise);
    CAP_PLUGIN_METHOD(getGmailMessageDetails, CAPPluginReturnPromise);
    CAP_PLUGIN_METHOD(getGmailAttachment, CAPPluginReturnPromise);
)
```

```
// ios/App/App/Plugins/SmsReader/SmsReaderPlugin.m

#import <Capacitor/Capacitor.h>

CAP_PLUGIN(SmsReaderPlugin, "SmsReader",
    CAP_PLUGIN_METHOD(checkPermissions, CAPPluginReturnPromise);
    CAP_PLUGIN_METHOD(requestPermissions, CAPPluginReturnPromise);
)
```

```
        CAP_PLUGIN_METHOD(getMessages, CAPPluginReturnPromise);  
    )
```

```
// ios/App/App/Plugins/PdfParser/PdfParserPlugin.m  
  
#import <Capacitor/Capacitor.h>  
  
CAP_PLUGIN(PdfParserPlugin, "PdfParser",  
    CAP_PLUGIN_METHOD(parsePdfText, CAPPluginReturnPromise);  
)
```

```
// ios/App/App/Plugins/Ocr/OcrPlugin.m  
  
#import <Capacitor/Capacitor.h>  
  
CAP_PLUGIN(OcrPlugin, "Ocr",  
    CAP_PLUGIN_METHOD(recognizeText, CAPPluginReturnPromise);  
)
```

```
// ios/App/App/Plugins/SecureStorage/SecureStoragePlugin.m  
  
#import <Capacitor/Capacitor.h>  
  
CAP_PLUGIN(SecureStoragePlugin, "SecureStorage",  
    CAP_PLUGIN_METHOD(encryptString, CAPPluginReturnPromise);  
    CAP_PLUGIN_METHOD(decryptString, CAPPluginReturnPromise);  
)
```

5. Google Cloud Console iOS Yapılandırması

5.1 iOS OAuth 2.0 Client ID Oluşturma

1. **Google Cloud Console'a gidin:** <https://console.cloud.google.com>
2. **APIs & Services > Credentials** bölümüne gidin
3. **Create Credentials > OAuth 2.0 Client ID** seçin
4. **Application type:** iOS seçin
5. **Bundle ID:** `com.codeyzer.ekstre` girin
6. **App Store ID:** (Henüz yoksa boş bırakın)
7. **Team ID:** Apple Developer hesabınızdan alın (örn: `ABCD1234EF`)

5.2 Firebase iOS Yapılandırması

1. **Firebase Console'a gidin:** <https://console.firebase.google.com>

2. **Projenizi seçin** (Android ile aynı proje)
 3. **Add app > iOS** seçin
 4. **Bundle ID:** `com.codeyzer.ekstre`
 5. **GoogleService-Info.plist** dosyasını indirin
 6. `ios/App/App/` klasörüne kopyalayın
-

6. TypeScript Platform Kontrolleri

6.1 Platform-Specific Service Updates

```
// src/services/sms-parsing/sms-processor.ts

import { Capacitor } from '@capacitor/core';

export class StatementProcessor {

  async fetchAndParseStatements(): Promise<ParsedStatement[]> {
    const platform = Capacitor.getPlatform();

    // iOS'ta SMS okuma mevcut değil
    if (platform === 'ios') {
      console.log('[StatementProcessor] iOS platform detected - SMS
reading not available');
      return this.fetchEmailStatementsOnly();
    }

    // Android'de hem SMS hem Email
    const [smsStatements, emailStatements] = await Promise.all([
      this.fetchSmsStatements(),
      this.fetchEmailStatements()
    ]);

    return this.mergeAndDeduplicateStatements(smsStatements,
emailStatements);
  }

  async checkSmsPermission(): Promise<boolean> {
    const platform = Capacitor.getPlatform();

    if (platform === 'ios') {
      // iOS'ta SMS izni kavramı yok
      return false;
    }

    const result = await SmsReader.checkPermissions();
    return result.readSms === 'granted';
  }

  async requestSmsPermission(): Promise<boolean> {
    const platform = Capacitor.getPlatform();
```

```

    if (platform === 'ios') {
      // iOS'ta izin istenemez
      console.warn('[StatementProcessor] SMS permission not available
on iOS');
      return false;
    }

    const result = await SmsReader.requestPermissions();
    return result.readSms === 'granted';
  }
}

```

6.2 UI Platform Indicators

```

// src/components/PermissionStatus.tsx

import { Capacitor } from '@capacitor/core';
import { IonChip, IonIcon, IonLabel } from '@ionic/react';
import { mailOutline, chatbubbleOutline, checkmarkCircle, closeCircle }
from 'ionicons/icons';

export const PermissionStatus: React.FC = () => {
  const platform = Capacitor.getPlatform();
  const isIOS = platform === 'ios';

  return (
    <div className="permission-status">
      {/* Email her zaman mevcuttur */}
      <IonChip color="success">
        <IonIcon icon={mailOutline} />
        <IonLabel>E-posta</IonLabel>
        <IonIcon icon={checkmarkCircle} />
      </IonChip>

      {/* SMS sadece Android'de */}
      {!isIOS && (
        <IonChip color={smsPermissionGranted ? 'success' :
'warning'}>
          <IonIcon icon={chatbubbleOutline} />
          <IonLabel>SMS</IonLabel>
          <IonIcon icon={smsPermissionGranted ? checkmarkCircle :
closeCircle} />
        </IonChip>
      )}

      {isIOS && (
        <IonChip color="medium">
          <IonIcon icon={chatbubbleOutline} />
          <IonLabel>SMS (iOS'ta mevcuttur değil)</IonLabel>
        </IonChip>
      )}
    </div>
  )
}

```

```
};  
);  
</div>
```

7. Test Stratejisi

7.1 iOS Simulator Testleri

```
# Simulator'da çalıştır  
npx cap run ios --target="iPhone 15 Pro"  
  
# Xcode'da aç  
npx cap open ios
```

7.2 Plugin Test Checklist

Plugin	Test Senaryosu	Beklenen Sonuç
GoogleAuth	Sign In	Kullanıcı bilgileri döner
GoogleAuth	Silent Sign In	Önceki oturum devam eder
GoogleAuth	Sign Out	Oturum temizlenir
GoogleAuth	Gmail Search	Mesaj listesi döner
GoogleAuth	Gmail Details	Mesaj detayları döner
GoogleAuth	Gmail Attachment	Base64 ek verisi döner
GoogleAuth	Calendar Create	Event ID döner
GoogleAuth	Calendar Search	eventFound boolean döner
SmsReader	Check Permission	denied döner (iOS)
SmsReader	Request Permission	Reject döner (iOS)
SmsReader	Get Messages	Reject döner (iOS)
PdfParser	Parse PDF	Metin döner
Ocr	Recognize Text	OCR sonucu döner
SecureStorage	Encrypt	Base64 string döner
SecureStorage	Decrypt	Orijinal string döner

7.3 Vitest Platform Mocks

```
// src/__tests__/platform-mocks.ts

import { vi } from 'vitest';

export const mockIOSPlatform = () => {
  vi.mock('@capacitor/core', () => ({
    Capacitor: {
      getPlatform: () => 'ios',
      isNativePlatform: () => true
    }
  }));
};

export const mockAndroidPlatform = () => {
  vi.mock('@capacitor/core', () => ({
    Capacitor: {
      getPlatform: () => 'android',
      isNativePlatform: () => true
    }
  }));
};
```

8. Geliştirme Adımları (Checklist)

Faz 1: Proje Kurulumu

- ☐ `npm install @capacitor/ios`
- ☐ `npx cap add ios`
- ☐ `npx cap sync ios`
- ☐ Xcode'da proje yapılandırması
- ☐ CocoaPods bağımlılıkları (`pod install`)
- ☐ Signing & Capabilities ayarları

Faz 2: Firebase & Google Yapılandırması

- ☐ Firebase iOS app ekleme
- ☐ GoogleService-Info.plist ekleme
- ☐ Google Cloud Console iOS Client ID oluşturma
- ☐ Info.plist URL schemes ekleme
- ☐ AppDelegate.swift güncellemeleri

Faz 3: Plugin Implementasyonları

- ☐ SecureStoragePlugin.swift (Keychain)
- ☐ GoogleAuthPlugin.swift (Google Sign-In)
- ☐ GmailHandler.swift (Gmail API)
- ☐ CalendarHandler.swift (Calendar API)
- ☐ PdfParserPlugin.swift (PDFKit)

- ☐ OcrPlugin.swift (Vision Framework)
- ☐ SmsReaderPlugin.swift (Platform rejection)

Faz 4: Objective-C Bridge Dosyaları

- ☐ GoogleAuthPlugin.m
- ☐ SecureStoragePlugin.m
- ☐ PdfParserPlugin.m
- ☐ OcrPlugin.m
- ☐ SmsReaderPlugin.m

Faz 5: TypeScript Güncellemeleri

- ☐ Platform detection eklemeleri
- ☐ iOS-specific UI güncellemeleri
- ☐ SMS unavailability handling

Faz 6: Test & Debug

- ☐ Simulator testleri
- ☐ Gerçek cihaz testleri
- ☐ OAuth flow testleri
- ☐ PDF parsing testleri
- ☐ OCR testleri

Faz 7: App Store Hazırlığı

- ☐ App Icons (iOS specific sizes)
- ☐ Launch Screen
- ☐ Privacy Policy URL
- ☐ App Store Connect yapılandırması
- ☐ TestFlight dağıtımı

9. Bilinen Kısıtlamalar

Özellik	Android	iOS	Notlar
SMS Okuma	✓	✗	iOS API kısıtlaması
Gmail API	✓	✓	Tam destek
Calendar API	✓	✓	Tam destek
PDF Parsing	✓	✓	Farklı kütüphaneler
OCR	✓	✓	Vision vs ML Kit
Secure Storage	✓	✓	Keystore vs Keychain
Deep Linking	✓	✓	Farklı yapılandırma
Push Notifications	✓	✓	APNs gerekli

10. Kaynaklar

Apple Documentation

- [Vision Framework](#)
- [PDFKit](#)
- [Keychain Services](#)
- [CryptoKit](#)

Google Documentation

- [Google Sign-In iOS](#)
- [Google API Client Library \(Objective-C/Swift\)](#)
- [Firebase Auth iOS](#)

Capacitor Documentation

- [Capacitor iOS](#)
- [Custom Native iOS Code](#)
- [Plugin Development](#)

Bu doküman, Codezyer Ekstre Takip uygulamasının iOS entegrasyonu için referans kaynağı olarak hazırlanmıştır.

Son güncelleme: Ocak 2026