COFFEETIME

Specialty Coffee Tracking Application

Technical Specification & Requirements Document

Version 1.0

January 18, 2026

Prepared for: Claude Code Implementation

Table of Contents

1. Executive Summary

1.1 Purpose

CoffeeTime is a Progressive Web Application (PWA) designed to track specialty coffee brewing sessions. The application enables users to log brewing parameters, manage their equipment inventory (grinders, brewers, coffees), and analyze brewing trends over time. The app features voice input for hands-free logging and AI-powered coffee recognition from photographs.

1.2 Technology Stack

Component	Technology
Frontend Framework	Next.js 14+ with React 18
UI Library	shadcn/ui with Tailwind CSS
Backend/Database	Firebase (Firestore, Auth, Storage)
Hosting	Firebase Hosting
AI Services	Claude API (Anthropic)
Voice Input	Web Speech API
Application Type	Progressive Web App (PWA)

1.3 Target Platform

Primary: Android devices via PWA installation. Secondary: iOS Safari, Desktop browsers (Chrome, Firefox, Edge). The application will be installable from the browser with full offline capability.

2. Project Overview

2.1 Problem Statement

Coffee enthusiasts who experiment with brewing parameters lack a convenient way to track their brews, identify successful combinations, and learn from their brewing history. Existing solutions are either too complex, lack mobile support, or don’t integrate AI capabilities for streamlined data entry.

2.2 Solution

CoffeeTime provides a mobile-first, voice-enabled brewing journal that learns from your brewing patterns and makes data entry effortless through AI-powered parsing and coffee recognition.

2.3 Key Features

Core functionality includes:

- • Voice-to-structured-data brew logging
- • Photo-based coffee recognition and data enrichment
- • Equipment inventory management (grinders, brewers, coffees)
- • Searchable brew history with filters
- • Visual analytics and trend charts
- • Offline-first with automatic sync
- • Dark mode, modern aesthetic

2.4 Design Philosophy

The application must look and feel like a professionally designed product from 2025—clean, minimal, fast, and delightful to use. It should impress a young professional with its attention to detail, smooth animations, and thoughtful UX. No generic “app builder” aesthetics.

3. Technical Architecture

3.1 Architecture Overview

The application follows a three-tier architecture: Presentation Layer (Next.js PWA), Service Layer (Firebase Cloud Functions + Claude API), and Data Layer (Firestore). All tiers communicate via secure HTTPS.

3.2 Frontend Architecture

3.2.1 Framework: Next.js 14+

Next.js provides server-side rendering capabilities, excellent developer experience, and first-class PWA support via next-pwa. The App Router will be used for routing with React Server Components where appropriate.

3.2.2 UI Components: shadcn/ui

shadcn/ui provides accessible, customizable components built on Radix UI primitives. Components will be styled using Tailwind CSS with a custom theme for the coffee aesthetic (warm tones, dark mode first).

3.2.3 State Management

React Context for global state (auth, theme). React Query (TanStack Query) for server state management and caching. Local component state with useState/useReducer for UI state.

3.3 Backend Architecture

3.3.1 Firebase Services

Service	Purpose
Firestore	NoSQL document database for all application data
Authentication	Google Sign-In for user authentication
Cloud Storage	Image storage for coffee photos
Cloud Functions	Serverless functions for AI integration
Hosting	Static hosting with global CDN

3.3.2 Cloud Functions

Firebase Cloud Functions (Node.js) will handle AI integration, keeping API keys secure on the server. Functions include: parseVoiceInput (text to structured brew data),

recognizeCoffee (image to coffee metadata), and enrichCoffeeData (web lookup for coffee details).

3.4 PWA Configuration

Feature	Implementation
Service Worker	next-pwa with Workbox for caching
Manifest	Web app manifest with icons, theme colors
Offline Storage	Firestore offline persistence enabled
Install Prompt	Custom install banner with deferred prompt
Push Notifications	Firebase Cloud Messaging (future phase)

4. Data Model

4.1 Design Principles

All fields except required identifiers should be nullable to accommodate incomplete data. The application must gracefully handle missing values in both display and analytics. Foreign key relationships are implemented via document IDs stored as strings.

4.2 Collections

4.2.1 Users Collection

Path: /users/{userId}

Field	Type	Required	Description
id	string	Yes	Firebase Auth UID
email	string	Yes	User email from Google Auth
displayName	string	No	User display name
photoURL	string	No	Profile photo URL
createdAt	timestamp	Yes	Account creation time
updatedAt	timestamp	Yes	Last update time
preferences	map	No	User preferences (theme, units, etc.)

4.2.2 Coffees Collection

Path: /users/{userId}/coffees/{coffeeId}

Field	Type	Required	Description
id	string	Yes	Auto-generated document ID
name	string	Yes	Coffee name (e.g., “Geometry”)
roaster	string	No	Roaster name (e.g., “Onyx”)
origin	string	No	Country/region of origin
region	string	No	Specific region within country
farm	string	No	Farm or producer name

process	string	No	Processing method (washed, natural, etc.)
variety	string	No	Coffee variety (Gesha, Bourbon, etc.)
elevation	number	No	Growing elevation in meters
roastLevel	string	No	Light, medium, dark, etc.
roastDate	timestamp	No	Date coffee was roasted
flavorNotes	array	No	Roaster’s flavor descriptors
photoURL	string	No	Photo of coffee bag
notes	string	No	User notes about this coffee
isActive	boolean	Yes	Whether coffee appears in dropdowns
createdAt	timestamp	Yes	Record creation time
updatedAt	timestamp	Yes	Last update time

4.2.3 Grinders Collection

Path: /users/{userId}/grinders/{grinderId}

Field	Type	Required	Description
id	string	Yes	Auto-generated document ID
name	string	Yes	Grinder name (e.g., “Fellow Ode”)
brand	string	No	Manufacturer
model	string	No	Model number/name
type	string	No	Electric or hand
burrType	string	No	Flat, conical, etc.
burrSize	number	No	Burr diameter in mm
settingsMin	number	No	Minimum grind setting
settingsMax	number	No	Maximum grind setting
settingsType	string	No	Stepped or stepless
notes	string	No	User notes

isActive	boolean	Yes	Appears in dropdowns
createdAt	timestamp	Yes	Record creation time
updatedAt	timestamp	Yes	Last update time

4.2.4 Brewers Collection

Path: /users/{userId}/brewers/{brewerId}

Field	Type	Required	Description
id	string	Yes	Auto-generated document ID
name	string	Yes	Brewer name (e.g., “V60 02”)
brand	string	No	Manufacturer
type	string	No	Pour over, immersion, espresso, etc.
material	string	No	Plastic, ceramic, glass, metal
capacityMl	number	No	Maximum capacity in milliliters
filterType	string	No	Paper, metal, cloth
notes	string	No	User notes
isActive	boolean	Yes	Appears in dropdowns
createdAt	timestamp	Yes	Record creation time
updatedAt	timestamp	Yes	Last update time

4.2.5 BrewLogs Collection

Path: /users/{userId}/brewLogs/{brewLogId}

This is the primary collection for tracking individual brew sessions.

Field	Type	Required	Description
id	string	Yes	Auto-generated document ID
timestamp	timestamp	Yes	When the brew was made
coffeeId	string	No	Reference to coffee document
coffeeName	string	No	Denormalized coffee name for queries
grinderId	string	No	Reference to grinder document
grinderName	string	No	Denormalized grinder name
brewerId	string	No	Reference to brewer document
brewerName	string	No	Denormalized brewer name

doseGrams	number	No	Coffee dose in grams
waterGrams	number	No	Total water in grams
ratio	string	No	Calculated ratio (e.g., “1:16”)
waterTempF	number	No	Water temperature in Fahrenheit
waterTempC	number	No	Water temperature in Celsius
grindSetting	string	No	Grind setting used
bloomTimeSeconds	number	No	Bloom duration in seconds
bloomWaterGrams	number	No	Water used for bloom
totalTimeSeconds	number	No	Total brew time in seconds
techniqueNotes	string	No	Description of pour technique
tastingNotes	string	No	Flavor observations
rating	number	No	Rating from 1-10
photoURL	string	No	Photo of the brew
rawVoiceInput	string	No	Original voice transcription
createdAt	timestamp	Yes	Record creation time
updatedAt	timestamp	Yes	Last update time

4.2.6 SharedBrews Collection (Public Links)

Path: /sharedBrews/{shareId}

Public collection for shareable brew links. Anyone with the link can view.

Field	Type	Required	Description
id	string	Yes	Short unique share ID
ownerId	string	Yes	User who shared the brew
ownerDisplayName	string	No	Sharer’s display name
originalBrewId	string	Yes	Reference to original brew
brewData	map	Yes	Complete copy of brew data

expiresAt	timestamp	No	Optional expiration date
viewCount	number	Yes	Number of times viewed
createdAt	timestamp	Yes	When share was created

4.2.7 UserProfiles Collection (Follow System)

Path: /userProfiles/{userId}

Public profile data for the follow/subscribe system.

Field	Type	Required	Description
id	string	Yes	Same as Firebase Auth UID
displayName	string	Yes	Public display name
photoURL	string	No	Profile photo
shareCode	string	Yes	Short code for sharing (e.g., “COFFEE42”)
bio	string	No	Short bio or description
isPublic	boolean	Yes	Whether profile is discoverable
followerCount	number	Yes	Number of followers
followingCount	number	Yes	Number following
brewCount	number	Yes	Total public brews
createdAt	timestamp	Yes	Profile creation time

4.2.8 Follows Collection

Path: /userProfiles/{userId}/followers/{followerId}

Path: /userProfiles/{userId}/following/{followingId}

Bidirectional follow relationships stored as subcollections.

Field	Type	Required	Description
userId	string	Yes	The follower/following user ID
displayName	string	Yes	Denormalized display name
photoURL	string	No	Denormalized photo
followedAt	timestamp	Yes	When follow occurred

4.2.9 Circles Collection (Collaborative Groups)

Path: /circles/{circleId}

Shared groups where multiple users can view/contribute brews.

Field	Type	Required	Description
id	string	Yes	Auto-generated circle ID
name	string	Yes	Circle name (e.g., “Office Coffee Club”)
description	string	No	Circle description
photoURL	string	No	Circle cover image
ownerId	string	Yes	Creator/admin of circle
inviteCode	string	Yes	Code to join circle
memberCount	number	Yes	Number of members
isPublic	boolean	Yes	Whether circle is discoverable
createdAt	timestamp	Yes	Creation time
updatedAt	timestamp	Yes	Last activity time

4.2.10 Circle Members Subcollection

Path: /circles/{circleId}/members/{userId}

Field	Type	Required	Description
userId	string	Yes	Member’s user ID
displayName	string	Yes	Denormalized name
photoURL	string	No	Denormalized photo
role	string	Yes	admin, contributor, or viewer
joinedAt	timestamp	Yes	When user joined

4.2.11 Circle Brews Subcollection

Path: /circles/{circleId}/brews/{brewId}

Brews posted to a circle. Same structure as BrewLog plus contributor info.

Field	Type	Required	Description

...brewLogFields	...	...	All fields from BrewLog
contributorId	string	Yes	User who posted this brew
contributorName	string	Yes	Denormalized contributor name
contributorPhotoURL	string	No	Denormalized photo

4.2.12 DriveExports Collection

Path: /users/{userId}/driveExports/{exportId}

Tracks Google Drive export history.

Field	Type	Required	Description
id	string	Yes	Export record ID
driveFileId	string	Yes	Google Drive file ID
fileName	string	Yes	Name of exported file
fileType	string	Yes	csv, json, or sheets
brewCount	number	Yes	Number of brews exported
dateRange	map	No	Start/end dates if filtered
exportedAt	timestamp	Yes	When export occurred
driveLink	string	Yes	Shareable Drive link

4.3 Firestore Indexes

Composite indexes required for common queries:

Index	Fields
Brew history by date	userId ASC, timestamp DESC
Brews by coffee	userId ASC, coffeeId ASC, timestamp DESC
Brews by rating	userId ASC, rating DESC, timestamp DESC
Brews by brewer	userId ASC, brewerId ASC, timestamp DESC

5. User Interface Specification

5.1 Design System

5.1.1 Color Palette

Dark mode first, with warm coffee-inspired accent colors.

Color	Value / Usage
Background (dark)	#0A0A0B - Primary background
Surface (dark)	#141416 - Cards, elevated surfaces
Surface Hover	#1C1C1F - Interactive states
Border	#2A2A2E - Subtle dividers
Text Primary	#FAFAFA - Main content
Text Secondary	#A1A1AA - Labels, hints
Accent Primary	#D4A574 - Warm brown, CTAs
Accent Secondary	#8B7355 - Secondary actions
Success	#4ADE80 - Positive feedback
Warning	#FBBF24 - Warnings
Error	#F87171 - Errors

5.1.2 Typography

Element	Specification
Font Family	Inter (primary), system-ui fallback
Heading 1	32px / 600 weight / -0.02em tracking
Heading 2	24px / 600 weight / -0.01em tracking
Heading 3	18px / 600 weight
Body	16px / 400 weight / 1.5 line height
Small	14px / 400 weight
Caption	12px / 500 weight / uppercase optional

5.1.3 Spacing & Layout

Base unit: 4px. Common spacings: 8px (sm), 16px (md), 24px (lg), 32px (xl). Border radius: 8px for cards, 6px for buttons, 4px for inputs. Maximum content width: 480px (mobile), 768px (tablet).

5.1.4 Animation

Use Framer Motion for micro-interactions. Default duration: 200ms. Easing: cubic-bezier(0.4, 0, 0.2, 1). Page transitions: fade with subtle scale. List items: staggered fade-in.

5.2 Screen Specifications

5.2.1 Navigation Structure

Bottom navigation bar with four primary destinations: Log (home), History, Analytics, and Settings. The Log screen is the default landing page.

5.2.2 Log Brew Screen (Home)

Primary screen for recording brew sessions.

Header: Date/time display, current coffee name if selected

Voice Input Button: Large, prominent microphone FAB. Tap to start recording, tap again to stop. Visual feedback during recording (pulse animation). Transcription appears in real-time.

Quick Entry Cards: Expandable sections for each parameter group: Equipment (coffee, grinder, brewer dropdowns), Parameters (dose, water, temp, grind), Timing (bloom time/water, total time), Notes (technique, tasting, rating).

Save Button: Fixed at bottom, disabled until minimum required data entered.

5.2.3 History Screen

Searchable, filterable list of past brews.

Search Bar: Full-text search across coffee names, notes, roasters

Filter Chips: Quick filters: Date range, Coffee, Brewer, Rating (4+, 7+)

Brew Cards: Each card shows: Date/time, Coffee name, Brewer, Key params (dose, ratio, time), Rating (if exists), Thumbnail (if photo exists). Tap to view full details or edit.

Empty State: Friendly illustration with prompt to log first brew

5.2.4 Analytics Screen

Visual insights from brewing data.

Summary Stats: Total brews, Average rating, Most used coffee, Most used brewer

Charts (use Recharts):

- • Brews over time (line/bar chart, daily/weekly/monthly)
- • Rating distribution (histogram)
- • Ratio vs Rating scatter plot
- • Brews by coffee (pie/donut chart)
- • Temperature vs Rating correlation

5.2.5 Settings/Manage Screen

Equipment management and app settings.

Equipment Sections:

- • My Coffees - List with add/edit/archive
- • My Grinders - List with add/edit/archive
- • My Brewers - List with add/edit/archive

Preferences: Temperature units (F/C), Default values, Theme toggle

Account: Profile info, Sign out, Export data

About: Version, Credits

5.2.6 Add/Edit Coffee Modal

Full-screen modal for coffee management.

Photo Capture: Camera button to photograph coffee bag. AI recognition button to auto-fill fields from photo.

Form Fields: All fields from Coffee data model, organized in logical groups. Roast date picker. Flavor notes as tag input.

6. Feature Requirements

6.1 Authentication

Requirement	Details
Sign-in Method	Google Sign-In only (simplicity)
Session Persistence	Remember user across app restarts
Sign Out	Clear local data, return to sign-in screen
Account Deletion	Future: Request deletion via support

6.2 Brew Logging

Requirement	Details
Voice Input	Tap microphone, speak naturally, AI parses to fields
Manual Entry	All fields editable via form inputs
Quick Log	Save with minimal data (just timestamp valid)
Edit Existing	Full edit capability for any past brew
Delete Brew	Confirmation dialog before deletion
Duplicate Brew	Copy previous brew as starting point

6.3 Equipment Management

Requirement	Details
Add Equipment	Forms for coffee, grinder, brewer
Photo Upload	Camera or gallery for coffee photos
AI Recognition	Extract coffee details from bag photo
Archive vs Delete	Soft delete keeps history, hides from dropdowns
Sort/Filter	Alphabetical, most used, recently added

6.4 History & Search

Requirement	Details

Infinite Scroll	Load more as user scrolls
Search	Full-text search on coffee, roaster, notes
Filters	Date range, coffee, brewer, grinder, rating
Sort Options	Newest, oldest, highest rated
Detail View	Full brew details with edit option

6.5 Analytics

Requirement	Details
Date Range Selector	Last 7 days, 30 days, 90 days, all time
Summary Statistics	Calculated from filtered data
Interactive Charts	Tap for details, zoom/pan where appropriate
Export	Future: Export data as CSV

6.6 Offline Capability

Requirement	Details
Offline Reads	All data cached locally via Firestore
Offline Writes	Queue changes, sync when online
Sync Indicator	Show pending sync count if offline
Conflict Resolution	Last-write-wins (Firestore default)

7. AI Integration

7.1 Voice Input Parsing

Flow:

1. User taps microphone button
2. Web Speech API transcribes audio to text
3. Text sent to Cloud Function
4. Cloud Function calls Claude API with parsing prompt
5. Structured JSON returned to client
6. Form fields populated, user confirms/edits

7.1.1 Example Voice Input

```
"Made a V60 this morning with 18 grams of Onyx Geometry,
300 grams of water at 205 degrees, ground on 5 on the Ode,
3 minute brew time, really fruity and sweet, I'd give it an 8"
```

7.1.2 Expected Output

```
{
  "brewerId": null,
  "brewerName": "V60",
  "coffeeId": null,
  "coffeeName": "Onyx Geometry",
  "grinderId": null,
  "grinderName": "Ode",
  "doseGrams": 18,
  "waterGrams": 300,
  "waterTempF": 205,
  "grindSetting": "5",
  "totalTimeSeconds": 180,
  "tastingNotes": "fruity and sweet",
  "rating": 8
}
```

7.2 Coffee Photo Recognition

Flow:

1. User photographs coffee bag
2. Image uploaded to Cloud Storage
3. Cloud Function triggered
4. Claude Vision API analyzes image
5. Extracted: roaster, name, origin, process, variety, flavor notes
6. User reviews and confirms data

7.3 Data Enrichment

When user enters a coffee name (e.g., “Onyx Geometry”), a Cloud Function can optionally search for additional details and suggest: origin, process, variety, elevation, typical flavor profile. This is a “nice to have” feature for Phase 2.

7.4 Claude API Configuration

Setting	Value
Model	claude-sonnet-4-20250514 (cost-effective)
Max Tokens	1024 (parsing), 2048 (recognition)
Temperature	0.3 (deterministic for parsing)
API Key Storage	Firebase environment config

8. Security & Authentication

8.1 Firebase Security Rules

Firestore rules ensure proper access control for private and shared data:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // Private user data
    match /users/{userId} {
      allow read, write: if request.auth != null
                         && request.auth.uid == userId;

      match /{subcollection}/{docId} {
        allow read, write: if request.auth != null
                           && request.auth.uid == userId;
      }
    }

    // Public shared brews (anyone can read)
    match /sharedBrews/{shareId} {
      allow read: if true;
      allow create: if request.auth != null;
      allow update, delete: if request.auth != null
                            && request.auth.uid == resource.data.ownerId;
    }

    // Public user profiles
    match /userProfiles/{userId} {
      allow read: if true;
      allow write: if request.auth != null
                   && request.auth.uid == userId;

      // Followers subcollection
      match /followers/{followerId} {
        allow read: if true;
        allow create, delete: if request.auth != null
                              && request.auth.uid == followerId;
      }

      // Following subcollection
      match /following/{followingId} {
        allow read: if request.auth != null
                    && request.auth.uid == userId;
        allow write: if request.auth != null
                     && request.auth.uid == userId;
      }
    }

    // Circles (collaborative groups)
    match /circles/{circleId} {
      allow read: if resource.data.isPublic == true
                  || isMember(circleId);
      allow create: if request.auth != null;
```

```
      allow update, delete: if isCircleAdmin(circleId);

      match /members/{memberId} {
        allow read: if isMember(circleId);
        allow create: if request.auth != null;
        allow delete: if request.auth.uid == memberId
                      || isCircleAdmin(circleId);
      }

      match /brews/{brewId} {
        allow read: if isMember(circleId);
        allow create: if isContributor(circleId);
        allow update, delete: if request.auth.uid ==
resource.data.contributorId
                              || isCircleAdmin(circleId);
      }
    }

    // Helper functions
    function isMember(circleId) {
      return request.auth != null &&

exists(/databases/$(database)/documents/circles/$(circleId)/members/$(request.a
uth.uid));
    }

    function isCircleAdmin(circleId) {
      return request.auth != null &&

get(/databases/$(database)/documents/circles/$(circleId)/members/$(request.auth
.uid)).data.role == "admin";
    }

    function isContributor(circleId) {
      let member =
get(/databases/$(database)/documents/circles/$(circleId)/members/$(request.auth
.uid));
      return member.data.role == "admin" || member.data.role == "contributor";
    }
  }
}
```

8.2 Cloud Storage Rules

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /users/{userId}/{allPaths=**} {
      allow read, write: if request.auth != null
                         && request.auth.uid == userId;
    }

    // Shared brew images (public read)
    match /shared/{allPaths=**} {
      allow read: if true;
```

```
      allow write: if request.auth != null;
    }

    // Circle images
    match /circles/{circleId}/{allPaths=**} {
      allow read: if request.auth != null;
      allow write: if request.auth != null;
    }
  }
}
```

8.3 API Key Security

Claude API keys are stored in Firebase environment configuration, never exposed to the client. All AI calls route through Cloud Functions which authenticate the user before processing.

9. Implementation Phases

9.1 Phase 1: Foundation (MVP)

Goal: Working app with core functionality

- • Project setup (Next.js, Firebase, Tailwind, shadcn/ui)
- • Google authentication
- • Data model implementation in Firestore
- • Manual brew logging (form-based)
- • Basic history view
- • Equipment management (coffees, grinders, brewers)
- • PWA configuration and installability

9.2 Phase 2: AI Features

Goal: Voice input and photo recognition

- • Cloud Functions setup
- • Voice input with Web Speech API
- • Claude integration for voice parsing
- • Coffee photo capture and upload
- • Claude Vision integration for recognition

9.3 Phase 3: Analytics & Polish

Goal: Insights and refined UX

- • Analytics dashboard with charts
- • Advanced search and filters
- • Animations and micro-interactions
- • Offline indicator and sync status
- • Performance optimization

9.4 Phase 4: Share Individual Brews

Goal: Shareable links for individual brews

- • SharedBrews collection and security rules
- • Share button on brew detail screen
- • Generate short share codes
- • Public /shared/[id] route (no auth required)
- • Copy link and native share sheet

• View count tracking and optional expiration

9.5 Phase 5: Follow/Subscribe System

Goal: Follow other users and see their brews

• UserProfiles collection with share codes

• Profile settings screen (bio, public toggle)

• Follow/unfollow functionality

• Following feed showing followed users’ brews

• Followers/following lists

• User search by share code

9.6 Phase 6: Coffee Circles (Groups)

Goal: Collaborative groups for sharing brews

• Circles collection with membership management

• Create/edit circle with name, description, photo

• Invite members via code or link

• Role-based permissions (admin, contributor, viewer)

• Post brews to personal log AND/OR circle

• Circle feed with all members’ contributions

• Leave circle and admin transfer

9.7 Phase 7: Google Drive Export

Goal: Export data to Google Drive for sharing/backup

• Google Drive API integration

• OAuth consent for Drive access

• Export as CSV, JSON, or Google Sheet

• Date range and filter options for export

• Export history with Drive links

• Share exported files via Drive sharing

9.8 Phase 8: Future Enhancements

• Import from Google Drive/CSV

• Coffee inventory tracking

• Brew recommendations based on history

• Public circle discovery

• Apple Watch companion

10. File Structure

Recommended project structure:

```
coffeetime/
├── app/
│   ├── (auth)/
│   │   └── login/
│   │       └── page.tsx
│   ├── (main)/
│   │   ├── layout.tsx          # Bottom nav layout
│   │   ├── page.tsx            # Log brew (home)
│   │   ├── history/
│   │   │   └── page.tsx
│   │   ├── analytics/
│   │   │   └── page.tsx
│   │   └── settings/
│   │       └── page.tsx
│   ├── layout.tsx              # Root layout
│   ├── globals.css
│   └── manifest.ts             # PWA manifest
├── components/
│   ├── ui/                     # shadcn components
│   ├── brew/
│   │   ├── BrewForm.tsx
│   │   ├── BrewCard.tsx
│   │   ├── VoiceInput.tsx
│   │   └── ParameterInput.tsx
│   ├── equipment/
│   │   ├── CoffeeForm.tsx
│   │   ├── GrinderForm.tsx
│   │   ├── BrewerForm.tsx
│   │   └── EquipmentList.tsx
│   ├── analytics/
│   │   ├── BrewsOverTime.tsx
│   │   ├── RatingDistribution.tsx
│   │   └── StatsSummary.tsx
│   └── common/
│       ├── BottomNav.tsx
│       ├── Header.tsx
│       └── LoadingSpinner.tsx
├── lib/
│   ├── firebase/
│   │   ├── config.ts
│   │   ├── auth.ts
│   │   └── firestore.ts
│   ├── hooks/
│   │   ├── useAuth.ts
│   │   ├── useCoffeeTimes.ts
│   │   ├── useCoffees.ts
│   │   ├── useGrinders.ts
│   │   └── useBrewers.ts
│   ├── services/
│   │   ├── brewService.ts
│   │   ├── coffeeService.ts
│   │   ├── grinderService.ts
│   │   ├── brewerService.ts
```

```
│   │   └── aiService.ts
│   ├── types/
│   │   └── index.ts            # TypeScript interfaces
│   └── utils/
│       ├── formatters.ts
│       └── validators.ts
├── functions/                   # Firebase Cloud Functions
│   ├── src/
│   │   ├── index.ts
│   │   ├── parseVoiceInput.ts
│   │   └── recognizeCoffee.ts
│   ├── package.json
│   └── tsconfig.json
├── public/
│   ├── icons/
│   └── images/
├── .env.local
├── .env.example
├── firebase.json
├── firestore.rules
├── storage.rules
├── next.config.js
├── tailwind.config.js
├── tsconfig.json
└── package.json
```

11. Claude Code Prompt

The following prompt can be used with Claude Code to begin implementation. Copy and paste this entire section as your initial prompt.

CLAUDE CODE IMPLEMENTATION PROMPT

```
# CoffeeTime - Specialty Coffee Tracking PWA

## Project Overview
Build a Progressive Web App called "CoffeeTime" for tracking specialty
coffee brewing sessions. The app targets Android devices (PWA installable)
with offline capability.

## Tech Stack
- Frontend: Next.js 14+ with App Router
- UI: shadcn/ui + Tailwind CSS
- Backend: Firebase (Firestore, Auth, Storage, Cloud Functions)
- AI: Claude API (via Cloud Functions)
- Voice: Web Speech API

## Design Requirements
- Dark mode first with warm coffee accent colors (#D4A574)
- Modern, minimal aesthetic suitable for 2025
- Smooth animations (Framer Motion)
- Mobile-first, max content width 480px

## Core Features (Phase 1 - MVP)
1. Google Sign-In authentication
2. Manual brew logging with form inputs
3. Equipment management (coffees, grinders, brewers)
4. Brew history with search and filters
5. PWA configuration (installable, works offline)

## Data Model
Collections under /users/{userId}/:
- coffees: name*, roaster, origin, process, variety, roastDate, etc.
- grinders: name*, brand, type, burrType, settingsMin/Max, etc.
- brewers: name*, brand, type, material, capacityMl, etc.
- brewLogs: timestamp*, coffeeId, grinderId, brewerId, doseGrams,
            waterGrams, waterTempF, grindSetting, totalTimeSeconds,
            tastingNotes, rating, etc.

(* = required, all other fields nullable)

## Key Screens
1. Log Brew (home): Form with dropdowns for equipment, parameter inputs
2. History: Searchable list of past brews with cards
3. Analytics: Charts showing trends (Phase 3)
4. Settings: Equipment management, preferences, account

## Implementation Order
1. Initialize Next.js project with TypeScript
2. Configure Tailwind and install shadcn/ui
```

```
3. Set up Firebase project and configuration
4. Implement authentication flow
5. Create TypeScript interfaces for data model
6. Build Firestore service layer (CRUD operations)
7. Create equipment management screens
8. Build brew logging form
9. Implement history view
10. Configure PWA (next-pwa)
11. Add offline persistence
12. Polish UI and add animations

## Important Notes
- All fields except IDs and timestamps should allow null/undefined
- Denormalize names (coffeeName, etc.) in brewLogs for query efficiency
- Use React Query for server state management
- Enable Firestore offline persistence
- Handle incomplete records gracefully in UI

Please start by setting up the project foundation: Next.js with
TypeScript, Tailwind CSS, and shadcn/ui. Then create the Firebase
configuration files and authentication flow.
```

— End of Document —