Prevently.ai - Complete Technical Specification

Table of Contents

1. Project Overview

App Description and Vision

Prevently.ai is a comprehensive preventive healthcare platform that leverages AI to provide personalized health insights, medication management, and family health coordination. The app serves as a digital health twin, tracking user vitals, providing AI-generated health tips, and facilitating proactive health management.

Target Audience

- Health-conscious individuals aged 25-65
- Families seeking coordinated health management
- Patients with chronic conditions requiring medication tracking
- Users interested in preventive healthcare and wellness optimization

Core Value Proposition

- **AI-Powered Personalization**: Custom health tips based on user profile and health data
- **Digital Health Twin**: Comprehensive health modeling and risk assessment
- **Family Health Hub**: Coordinated health management for family members
- **Preventive Focus**: Emphasis on prevention rather than treatment
- **Professional UI/UX**: Liquid glass aesthetic with neumorphic design elements

2. Technical Architecture

Tech Stack with Versions

Frontend

- **Next.js**: 14.2.28 (App Router)
- **React**: 18.2.0
- **TypeScript**: 5.2.2
- **Tailwind CSS**: 3.3.3
- **Framer Motion**: 12.23.0 (animations)
- **Radix UI**: Various components (2.x.x)

Backend & Database

- **Prisma ORM**: 6.7.0
- **PostgreSQL**: Database
- **NextAuth.js**: 4.24.11 (authentication)
- **bcryptjs**: 2.4.3 (password hashing)

UI Components

- **Radix UI Primitives**: Complete component library
- **Lucide React**: 0.446.0 (icons)
- **React Hook Form**: 7.53.0 (form management)
- **Zod**: 3.23.8 (validation)

Charts & Visualization

- **Chart.js**: 4.4.9
- **React Chart.js 2**: 5.3.0
- **Plotly.js**: 2.35.3
- **React Plotly.js**: 2.6.0
- **Recharts**: 3.0.2

Additional Libraries

- **Date-fns**: 3.6.0 (date manipulation)
- **React Hot Toast**: 2.4.1 (notifications)
- **Zustand**: 5.0.3 (state management)
- **SWR**: 2.2.4 (data fetching)

Project Structure

```
app/
├── app/                    # Next.js App Router
│   ├── api/                # API routes
│   │   ├── auth/           # Authentication endpoints
│   │   ├── digital-twin/   # Digital twin data
│   │   ├── family/         # Family management
│   │   ├── insights/       # AI insights
│   │   ├── medications/    # Medication tracking
│   │   ├── onboarding/     # User onboarding
│   │   ├── surveys/        # Health surveys
│   │   ├── tips/           # Health tips
│   │   └── user/           # User management
│   ├── dashboard/          # Dashboard page
│   ├── digital-twin/       # Digital twin visualization
│   ├── family/             # Family management
│   ├── knowledge-bank/     # Health knowledge base
│   ├── medications/        # Medication management
│   ├── onboarding/         # User onboarding flow
│   ├── settings/           # User settings
│   ├── tips/               # Health tips browser
│   ├── globals.css         # Global styles
│   ├── layout.tsx          # Root layout
│   └── page.tsx            # Landing/auth page
├── components/             # React components
│   ├── auth/               # Authentication components
│   ├── dashboard/          # Dashboard-specific components
│   ├── digital-twin/       # Digital twin components
│   ├── family/             # Family management components
│   ├── knowledge-bank/     # Knowledge base components
│   ├── layout/             # Layout components
│   ├── medications/        # Medication components
│   ├── onboarding/         # Onboarding wizard
│   ├── providers/          # Context providers
│   ├── settings/           # Settings components
│   ├── tips/               # Tips components
│   └── ui/                 # Reusable UI components
├── lib/                    # Utility libraries
│   ├── auth.ts             # NextAuth configuration
│   ├── db.ts               # Prisma client
│   ├── types.ts            # TypeScript types
│   └── utils.ts            # Utility functions
├── prisma/                 # Database schema
│   └── schema.prisma       # Prisma schema
├── scripts/                # Database scripts
│   └── seed.ts             # Database seeding
└── public/                 # Static assets
    └── logo.svg            # App logo
```

Database Schema

Core Tables

Users Table

```
model User {
  id                        String    @id @default(cuid())
  name                      String?
  email                     String    @unique
  emailVerified             DateTime?
  image                     String?
  password                  String?

  // Onboarding fields
  onboardingCompleted       Boolean   @default(false)
  dateOfBirth               DateTime?
  gender                    String?
  height                    Float?
  weight                    Float?
  activityLevel             String?
  healthGoals               String[]
  medicalConditions         String[]
  allergies                 String[]
  currentMedications        String[]

  // Preferences
  notificationsEnabled      Boolean   @default(true)
  weeklyReportsEnabled      Boolean   @default(true)
  familySharingEnabled      Boolean   @default(false)
  aiPersonalizationLevel    String    @default("balanced")
  dataRetentionPreference   String    @default("standard")

  // Relations
  accounts                  Account[]
  sessions                  Session[]
  healthTips                HealthTip[]
  medications               Medication[]
  familyMembers             FamilyMember[]
  digitalTwinData           DigitalTwinData[]
  surveyResponses           SurveyResponse[]
  accountabilityPartnerships AccountabilityPartnership[]

  createdAt                 DateTime  @default(now())
  updatedAt                 DateTime  @updatedAt
}
```

HealthTip Table

```
model HealthTip {
  id                String   @id @default(cuid())
  userId            String
  user              User     @relation(fields: [userId], references: [id])

  title             String
  content           String   @db.Text
  category          String   // nutrition, exercise, mental_health, sleep, hydration, ge
neral
  priority          String   @default("medium") // low, medium, high, urgent

  isPersonalized    Boolean  @default(false)
  viewed            Boolean  @default(false)
  rating            Int?     // 1-5 star rating

  // AI and Expert Review
  aiGenerated       Boolean  @default(false)
  expertReviewed    Boolean  @default(false)
  evidenceLevel     String?  // Research quality indicator
  sourceUrls        String[] // Links to research/sources

  createdAt         DateTime @default(now())
  updatedAt         DateTime @updatedAt
}
```

Medication Table

```
model Medication {
  id             String   @id @default(cuid())
  userId         String
  user           User     @relation(fields: [userId], references: [id])

  name           String
  dosage         String
  frequency      String   // daily, twice_daily, weekly, as_needed
  timeOfDay      String[] // morning, afternoon, evening, night
  instructions   String?
  prescribedBy   String?

  startDate      DateTime
  endDate        DateTime?

  status         String   @default("active") // active, paused, completed, discontinued

  // Tracking
  adherenceRate Float?   // Percentage of doses taken as prescribed
  lastTaken      DateTime?
  nextDue        DateTime?

  createdAt      DateTime @default(now())
  updatedAt      DateTime @updatedAt
}
```

DigitalTwinData Table

```
model DigitalTwinData {
  id           String   @id @default(cuid())
  userId       String
  user         User     @relation(fields: [userId], references: [id])

  bodySystem   String   // cardiovascular, nervous, respiratory, digestive, immune, en-
docrine
  dataType     String   // Heart Rate, Blood Pressure, Stress Level, etc.
  value        Float
  unit         String

  riskLevel    String   @default("low") // low, moderate, high
  trend        String   @default("stable") // improving, stable, declining

  // Data source
  source       String?  // manual, device, lab_result, app_sync
  deviceId     String?

  // Metadata
  notes        String?
  isVerified   Boolean  @default(false)

  lastUpdated  DateTime @default(now())
  createdAt    DateTime @default(now())
}
```

API Endpoints

Authentication

- `POST /api/auth/signup` - User registration
- `POST /api/auth/[...nextauth]` - NextAuth.js handlers

Health Tips

- `POST /api/tips/generate` - Generate AI-powered health tip
- `POST /api/tips/rate` - Rate a health tip
- `POST /api/tips/generate-category` - Generate category-specific tip

Medications

- `GET /api/medications` - Get user medications
- `POST /api/medications` - Add new medication
- `PUT /api/medications/[id]` - Update medication
- `DELETE /api/medications/[id]` - Delete medication
- `POST /api/medications/log` - Log medication taken

Family Management

- `GET /api/family` - Get family members
- `POST /api/family` - Add family member
- `PUT /api/family/[id]` - Update family member
- `DELETE /api/family/[id]` - Delete family member
- `POST /api/family/accountability` - Create accountability partnership

Digital Twin

- `GET /api/digital-twin/data` - Get digital twin data
- `POST /api/digital-twin/data` - Add health data point

User Management

- `GET /api/user/profile` - Get user profile
- `PUT /api/user/profile` - Update user profile
- `POST /api/user/water-intake` - Log water intake
- `GET /api/user/settings` - Get user settings
- `PUT /api/user/settings` - Update user settings
- `POST /api/user/export` - Export user data
- `DELETE /api/user/delete` - Delete user account

Surveys

- `POST /api/surveys/complete` - Complete survey

Insights

- `POST /api/insights/feedback` - Provide feedback on AI insight

Onboarding

- `POST /api/onboarding` - Save onboarding data

---

3. Feature Specifications

Core Features

1. User Authentication

- **Demo Account**: john@doe.com / johndoe123
- **Secure Login**: bcrypt password hashing
- **Session Management**: JWT-based sessions via NextAuth.js
- **Registration Flow**: Email/password with validation

2. AI-Powered Health Tips

- **Personalized Generation**: Tips based on user profile, health data, and preferences
- **Categories**: Prevention, nutrition, exercise, mental health, sleep, hydration, general
- **Rating System**: 1-5 star rating for tip quality
- **Evidence-Based**: Tips include evidence level and source URLs
- **Fresh Content**: Daily generation of new personalized tips

3. Digital Health Twin

- **Body Systems Tracking**: Cardiovascular, respiratory, nervous, digestive, immune, endocrine
- **Risk Assessment**: Low, moderate, high risk levels
- **Trend Analysis**: Improving, stable, declining trends
- **Data Sources**: Manual entry, device integration, lab results
- **Visualization**: Interactive charts and health model

4. Medication Management

- **Comprehensive Tracking**: Name, dosage, frequency, timing
- **Adherence Monitoring**: Track missed doses and adherence rates
- **Smart Reminders**: Time-based medication alerts
- **Prescription Management**: Prescriber information and instructions
- **Status Tracking**: Active, paused, completed, discontinued

5. Family Health Hub

- **Family Member Profiles**: Name, relationship, age, health summary
- **Emergency Contacts**: Designated emergency contact system
- **Consent Management**: Privacy and data sharing controls
- **Accountability Partners**: Mutual health goal support
- **Sharing Levels**: Basic, detailed, full health data sharing

6. Interactive Health Surveys

- **Flo-Style Engagement**: Modern, engaging survey interface
- **Multiple Question Types**: Scale, multiple choice, text, boolean, slider
- **Daily Check-ins**: Mood, energy, sleep quality assessments
- **AI Insights**: Generated insights from survey responses
- **Progress Tracking**: Historical survey data and trends

7. Knowledge Bank

- **Clinical Studies**: Evidence-based health information
- **Expert Content**: Professionally reviewed health articles
- **Search Functionality**: Find relevant health information
- **Categorized Content**: Organized by health topics

8. Settings & Preferences

- **Dark Mode**: System-wide dark theme support
- **High Contrast**: Accessibility mode for better visibility
- **Notification Settings**: Granular notification controls
- **Privacy Controls**: Data retention and sharing preferences
- **AI Personalization**: Adjustable AI recommendation levels

User Flows

Onboarding Flow

1. **Personal Information**: Age, gender, height, weight, emergency contact
2. **Medical History**: Conditions, allergies, surgeries, family history
3. **Lifestyle Assessment**: Activity level, sleep, diet, stress
4. **Vital Signs**: Heart rate, blood pressure, BMI
5. **Health Goals**: Primary goals, timeline, motivation

Daily Usage Flow

1. **Dashboard Landing**: Health score, streak, today’s tip
2. **Tip Interaction**: Read, rate, generate new tips
3. **Health Data Entry**: Log vitals, symptoms, mood
4. **Medication Tracking**: Mark medications as taken
5. **Survey Completion**: Daily health check-ins

Family Management Flow

1. **Add Family Member**: Basic information and relationship
2. **Set Permissions**: Consent and sharing level configuration
3. **Create Accountability**: Set up mutual health reminders
4. **Monitor Progress**: View family health summaries

4. UI/UX Requirements

Design System

Liquid Glass Aesthetic

The app implements Apple’s liquid glass design language with the following characteristics:

Core Glass Effects

```css
.liquid-glass {
  background: rgba(255, 255, 255, 0.85);
  backdrop-filter: blur(20px) saturate(1.8);
  -webkit-backdrop-filter: blur(20px) saturate(1.8);
  border: 1px solid rgba(255, 255, 255, 0.3);
  box-shadow:
    0 8px 32px rgba(0, 0, 0, 0.1),
    inset 0 1px 0 rgba(255, 255, 255, 0.8),
    inset 0 -1px 0 rgba(255, 255, 255, 0.2);
}
```

Interactive Elements

```css
.liquid-button {
  background: rgba(255, 255, 255, 0.9);
  backdrop-filter: blur(12px) saturate(1.4);
  border-radius: 12px;
  transition: all 0.2s ease;
}

.liquid-button:hover {
  transform: translateY(-2px);
  box-shadow: 0 8px 40px rgba(0, 0, 0, 0.12);
}
```

Neumorphic Design Elements

- **Soft Shadows**: Subtle depth with multiple shadow layers
- **Rounded Corners**: 12-16px border radius for cards and buttons
- **Pressed States**: Inset shadows for active elements
- **Gradient Backgrounds**: Subtle gradients for depth

Color Scheme

Light Mode

```css
:root {
  --background: 0 0% 100%;
  --foreground: 240 10% 3.9%;
  --primary: 221.2 83.2% 53.3%;
  --secondary: 210 40% 96%;
  --muted: 210 40% 96%;
  --accent: 210 40% 96%;
  --border: 214.3 31.8% 91.4%;
}
```

Dark Mode

```css
.dark {
  --background: 240 10% 3.9%;
  --foreground: 0 0% 98%;
  --primary: 221.2 83.2% 53.3%;
  --secondary: 240 3.7% 15.9%;
  --muted: 240 3.7% 15.9%;
  --accent: 240 3.7% 15.9%;
  --border: 240 3.7% 15.9%;
}
```

High Contrast Mode

```css
.high-contrast {
  --background: 0 0% 0%;
  --foreground: 0 0% 100%;
  --primary: 0 0% 100%;
  --muted: 0 0% 20%;
}
```

Typography

- **Font Family**: System fonts (San Francisco, Segoe UI, Roboto)
- **Font Weights**: 400 (regular), 500 (medium), 600 (semibold), 700 (bold)
- **Font Sizes**:
- Headings: 2xl (24px), xl (20px), lg (18px)
- Body: base (16px), sm (14px)
- Captions: xs (12px)

Responsive Design

- **Mobile First**: Designed for mobile with desktop enhancements
- **Breakpoints**:
- sm: 640px
- md: 768px
- lg: 1024px
- xl: 1280px
- **Grid System**: CSS Grid and Flexbox for layouts
- **Touch Targets**: Minimum 44px for interactive elements

Animation System

- **Framer Motion**: Page transitions and component animations
- **Micro-interactions**: Hover states, button presses, loading states
- **Breathing Animation**: Subtle scale animations for emphasis
- **Shimmer Effects**: Loading state animations
- **Spring Physics**: Natural feeling animations

Accessibility Features

- **High Contrast Mode**: Enhanced visibility for users with visual impairments
- **Focus Management**: Clear focus indicators and logical tab order
- **Screen Reader Support**: Proper ARIA labels and semantic HTML
- **Keyboard Navigation**: Full keyboard accessibility
- **Color Contrast**: WCAG AA compliant color ratios

5. Component Architecture

Component Tree

```
App
├── Layout
│   ├── Navbar
│   ├── ThemeProvider
│   └── AuthProvider
├── Pages
│   ├── AuthPage
│   ├── Dashboard
│   │   ├── DashboardClient
│   │   ├── DailyTipCard
│   │   ├── VitalsCard
│   │   ├── StreakCard
│   │   ├── MedicationReminder
│   │   └── WaterIntakeIndicator
│   ├── DigitalTwin
│   │   └── TwinVisualization
│   ├── Tips
│   │   └── TipsClient
│   ├── Medications
│   │   └── MedicationsClient
│   ├── Family
│   │   └── FamilyClient
│   ├── KnowledgeBank
│   │   └── KnowledgeBankClient
│   ├── Settings
│   │   └── SettingsClient
│   └── Onboarding
│       ├── OnboardingWizard
│       ├── PersonalInfoStep
│       ├── MedicalHistoryStep
│       ├── LifestyleStep
│       ├── VitalsStep
│       └── GoalsStep
└── UI Components
    ├── NeumorphicCard
    ├── NeumorphicButton
    ├── NeumorphicInput
    ├── SurveyComponent
    ├── AIInsightCard
    ├── DataImportModal
    ├── BackButton
    └── Radix UI Components
```

Reusable Component Specifications

NeumorphicCard

```
interface NeumorphicCardProps {
  children: React.ReactNode;
  className?: string;
  onClick?: () => void;
  variant?: 'default' | 'pressed' | 'elevated';
}
```

NeumorphicButton

```
interface NeumorphicButtonProps {
  children: React.ReactNode;
  onClick?: () => void;
  variant?: 'default' | 'primary' | 'secondary' | 'ghost';
  size?: 'sm' | 'md' | 'lg';
  disabled?: boolean;
  loading?: boolean;
}
```

SurveyComponent

```
interface SurveyComponentProps {
  survey: Survey;
  onComplete: (answers: Record<string, any>) => void;
  onClose: () => void;
}
```

AIInsightCard

```
interface AIInsightCardProps {
  insight: AIInsight;
  onFeedback: (insightId: string, helpful: boolean) => void;
  className?: string;
}
```

State Management Patterns

Local State

- **useState**: Component-level state for UI interactions
- **useEffect**: Side effects and data fetching
- **useReducer**: Complex state logic (survey responses, onboarding)

Global State

- **Zustand**: Lightweight state management for user preferences
- **SWR**: Server state management and caching
- **React Context**: Theme and authentication state

Form Management

- **React Hook Form**: Form state and validation
- **Zod**: Schema validation
- **Formik**: Complex multi-step forms (onboarding)

---

6. Setup Instructions

Prerequisites

- Node.js 18.x or higher
- PostgreSQL database
- Yarn or npm package manager

Environment Variables

Create `.env.local` file:

```
# Database
DATABASE_URL="postgresql://username:password@localhost:5432/prevently_db"

# NextAuth.js
NEXTAUTH_URL="http://localhost:3000"
NEXTAUTH_SECRET="your-secret-key"

# AI Service
ABACUSAI_API_KEY="your-abacus-ai-api-key"

# Optional: External integrations
GOOGLE_CLIENT_ID="your-google-client-id"
GOOGLE_CLIENT_SECRET="your-google-client-secret"
```

Step-by-Step Setup

1. Clone and Install Dependencies

```
# Clone the repository
git clone <repository-url>
cd prevently-ai/app

# Install dependencies
yarn install
# or
npm install
```

2. Database Setup

```
# Generate Prisma client
npx prisma generate

# Run database migrations
npx prisma db push

# Seed the database with demo data
npx prisma db seed
```

3. Development Server

```
# Start development server
yarn dev
# or
npm run dev
```

4. Build for Production

```
# Build the application
yarn build
# or
npm run build

# Start production server
yarn start
# or
npm start
```

Database Seeding

The seed script creates:
- Demo user: john@doe.com / johndoe123
- Sample medications (Vitamin D3, Omega-3)
- Health tips across all categories
- Family members with different relationships
- Digital twin data for all body systems
- Sample surveys and responses
- Accountability partnerships

Development Tools

- **ESLint**: Code linting with Next.js configuration
- **TypeScript**: Strict type checking enabled
- **Prettier**: Code formatting (via ESLint plugin)
- **Prisma Studio**: Database GUI (`npx prisma studio`)

---

7. Critical Fixes Implemented

Authentication Issues

Problem: Black screen on login, session management failures
Solution:
- Implemented proper NextAuth.js configuration with JWT strategy
- Added session callbacks for user ID persistence
- Fixed credential provider validation
- Added proper error handling for auth failures

UI Visibility Problems

Problem: Components not rendering, styling conflicts
Solution:
- Fixed CSS class conflicts between Tailwind and custom styles
- Implemented proper z-index layering for modals and overlays
- Added fallback styles for unsupported backdrop-filter
- Fixed responsive design breakpoints

TypeScript Errors

Problem: Type mismatches, missing interfaces
Solution:
- Created comprehensive type definitions in `lib/types.ts`
- Fixed Prisma client type imports
- Added proper prop interfaces for all components
- Implemented strict TypeScript configuration

Performance Optimizations

Problem: Slow page loads, unnecessary re-renders
Solution:
- Implemented SWR for efficient data fetching and caching
- Added React.memo for expensive components
- Optimized image loading with Next.js Image component
- Implemented lazy loading for non-critical components

Database Connection Issues

Problem: Prisma client connection failures
Solution:
- Fixed Prisma schema configuration for production
- Added proper connection pooling
- Implemented database connection retry logic
- Added environment-specific database URLs

Mobile Responsiveness

Problem: Poor mobile experience, touch targets too small
Solution:
- Implemented mobile-first responsive design
- Increased touch target sizes to 44px minimum
- Added proper viewport meta tags
- Optimized animations for mobile performance

Accessibility Improvements

Problem: Poor screen reader support, keyboard navigation issues
Solution:
- Added proper ARIA labels and roles
- Implemented logical tab order
- Added high contrast mode support
- Fixed color contrast ratios for WCAG compliance

---

8. API Integration

AI Tips Generation System

Abacus.AI Integration

```
const response = await fetch('https://apps.abacus.ai/v1/chat/completions', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${process.env.ABACUSAI_API_KEY}`,
  },
  body: JSON.stringify({
    model: 'gpt-4.1-mini',
    messages: [{ role: 'user', content: prompt }],
    response_format: { type: "json_object" },
    temperature: 0.7,
    max_tokens: 500,
  }),
});
```

Personalization Algorithm

Tips are personalized based on:
- User age, gender, health conditions
- Current medications and allergies
- Lifestyle factors (activity level, sleep, stress)
- Previous tip ratings and preferences
- Digital twin health data

Content Quality Assurance

- Evidence-based recommendations only
- Source URL validation
- Expert review flags
- User rating feedback loop
- Content moderation for safety

External Service Integrations

Health Data Sources

- **Manual Entry**: User-input health data
- **Device Integration**: Fitness trackers, smart scales
- **Lab Results**: Import from healthcare providers
- **App Sync**: Integration with health apps

Notification Services

- **Email**: Weekly health reports
- **Push Notifications**: Medication reminders
- **SMS**: Emergency contact alerts
- **In-App**: Real-time health insights

Authentication Flow

NextAuth.js Configuration

```
export const authOptions: NextAuthOptions = {
  providers: [
    CredentialsProvider({
      name: 'credentials',
      credentials: {
        email: { label: 'Email', type: 'email' },
        password: { label: 'Password', type: 'password' },
      },
      async authorize(credentials) {
        // Validation and authentication logic
      },
    }),
  ],
  session: { strategy: 'jwt' },
  pages: { signIn: '/' },
  callbacks: {
    async jwt({ token, user }) {
      if (user) token.id = user.id;
      return token;
    },
    async session({ session, token }) {
      if (session.user) session.user.id = token.id as string;
      return session;
    },
  },
};
```

Data Validation and Error Handling

Input Validation

- **Zod Schemas**: Type-safe validation for all inputs
- **Sanitization**: XSS prevention and data cleaning
- **Rate Limiting**: API endpoint protection
- **CSRF Protection**: Built-in Next.js CSRF protection

Error Handling

```
try {
  // API operation
} catch (error) {
  console.error('Operation failed:', error);
  return NextResponse.json(
    { error: 'Operation failed' },
    { status: 500 }
  );
}
```

---

9. Testing Requirements

Unit Testing

- **Jest**: Testing framework for utility functions

- **React Testing Library**: Component testing
- **Coverage Target**: 80% code coverage minimum

Integration Testing

- **API Route Testing**: Test all endpoint functionality
- **Database Testing**: Prisma operations validation
- **Authentication Testing**: Login/logout flows

End-to-End Testing

- **Playwright**: Full user journey testing
- **Critical Paths**:
- User registration and onboarding
- Daily tip generation and rating
- Medication tracking workflow
- Family member management
- Survey completion flow

Performance Testing

- **Lighthouse**: Performance, accessibility, SEO scores
- **Load Testing**: API endpoint stress testing
- **Mobile Performance**: Touch responsiveness testing

User Acceptance Criteria

Dashboard Functionality

- [ ] User can view personalized health dashboard
- [ ] Daily tip displays with rating capability
- [ ] Health score and streak are visible
- [ ] Water intake tracking works correctly
- [ ] Medication reminders appear when due

Onboarding Process

- [ ] New user can complete 5-step onboarding
- [ ] All form validation works correctly
- [ ] User profile is created successfully
- [ ] Initial health data is saved

Family Management

- [ ] User can add family members
- [ ] Consent and sharing levels work
- [ ] Accountability partnerships function
- [ ] Emergency contact system works

AI Tips System

- [ ] Tips generate based on user profile
- [ ] Rating system functions correctly
- [ ] Fresh tips appear daily
- [ ] Evidence sources are provided

10. Deployment Specifications

Production Environment Setup

Hosting Requirements

- **Platform**: Vercel, Netlify, or AWS
- **Node.js**: 18.x or higher
- **Database**: PostgreSQL (managed service recommended)
- **CDN**: For static asset delivery

Environment Configuration

```
# Production Environment Variables
NODE_ENV=production
DATABASE_URL="postgresql://prod_user:password@prod_host:5432/prevently_prod"
NEXTAUTH_URL="https://prevently.ai"
NEXTAUTH_SECRET="production-secret-key"
ABACUSAI_API_KEY="production-api-key"
```

Build Configuration

```
// next.config.js
const nextConfig = {
  output: 'standalone',
  images: { unoptimized: true },
  eslint: { ignoreDuringBuilds: false },
  typescript: { ignoreBuildErrors: false },
};
```

Database Migration Strategy

1. **Backup**: Create database backup before deployment
2. **Migration**: Run `prisma db push` or `prisma migrate deploy`
3. **Seeding**: Optional production data seeding
4. **Verification**: Test database connectivity and queries

Monitoring and Logging

Application Monitoring

- **Error Tracking**: Sentry or similar service
- **Performance Monitoring**: Real user monitoring
- **Uptime Monitoring**: Health check endpoints
- **Database Monitoring**: Query performance tracking

Logging Strategy

- **Structured Logging**: JSON format for log aggregation
- **Log Levels**: Error, warn, info, debug
- **Sensitive Data**: Exclude PII from logs
- **Retention**: 30-day log retention policy

Security Considerations

Data Protection

- **Encryption**: All sensitive data encrypted at rest
- **HTTPS**: SSL/TLS for all communications
- **API Security**: Rate limiting and authentication
- **GDPR Compliance**: Data export and deletion capabilities

Privacy Controls

- **Data Minimization**: Collect only necessary data
- **Consent Management**: Granular privacy controls
- **Data Retention**: Configurable retention periods
- **Anonymization**: Option to anonymize health data

Backup and Recovery

Database Backups

- **Frequency**: Daily automated backups
- **Retention**: 30-day backup retention
- **Testing**: Monthly backup restoration tests
- **Geographic Distribution**: Multi-region backup storage

Disaster Recovery

- **RTO**: 4-hour recovery time objective
- **RPO**: 1-hour recovery point objective
- **Failover**: Automated failover procedures
- **Communication**: Incident response plan

Conclusion

This comprehensive specification provides everything needed to rebuild the Prevently.ai app in Cursor IDE. The application represents a modern, AI-powered healthcare platform with a focus on prevention, family coordination, and user engagement through beautiful UI/UX design.

Key implementation highlights:
- **Modern Tech Stack**: Next.js 14, React 18, TypeScript, Prisma
- **AI Integration**: Personalized health tips via Abacus.AI
- **Professional Design**: Liquid glass aesthetic with neumorphic elements
- **Comprehensive Features**: Digital twin, medication tracking, family hub
- **Production Ready**: Authentication, security, accessibility, performance

The app is currently functional and launch-ready, with all critical issues resolved and a complete feature set implemented.