ChefVoice

All-in-One Technical Package for HST (Updated MVP Code & Documentation)

Version: v2 (includes latest Manus export + updated documentation)
Date: January 05, 2026

Purpose: Provide a single, self-contained document describing the ChefVoice MVP, system structure, key flows, diagrams, and technical scope required to productionise the platform to a pilot-ready release aligned to FSAI guidelines.

Product	ChefVoice - Voice-first HACCP compliance system for professional kitchens
Primary inputs	Voice logging (STT/NLP) + OCR delivery invoice capture
Compliance basis	HSE HACCP parameters; alignment to FSAI Safe Catering Pack outputs
Current state	Working MVP + documented architecture + pilots lined up
Goal for partner	Production-grade system: deterministic rules engine, audit trail, reliability, pilot deployment

Executive Summary

ChefVoice is an AI-assisted HACCP compliance system designed for real kitchen workflows. The MVP focuses on two high-value capture modes: (1) voice-first logging of HACCP checks (temps, cleaning, corrective actions), and (2) OCR-based delivery invoice capture to eliminate manual entry. AI is used to assist extraction and UX, while compliance decisions remain deterministic and auditable (rules engine + explicit confirmations).

What this pack contains: updated MVP repository structure, core flows, diagrams, and technical documentation (README, API documentation, and prioritised TODO backlog) to support a productionisation plan.

Target outcome: a pilot-ready release suitable for deployment into Irish kitchens, producing inspection-ready records aligned to FSAI expectations, with strong audit trail and reliability.

Updated MVP Repository Structure

High-level tree (truncated for readability):

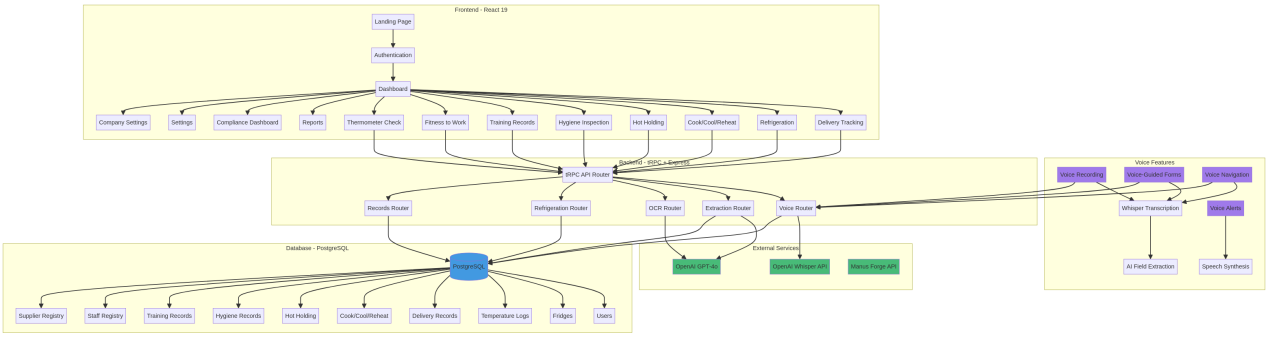
```
chefvoice_package/
  README.md
  diagrams/
    chefvoice_architecture.mmd
    chefvoice_architecture.png
    database_schema.mmd
    database_schema.png
    voice_features_flow.mmd
    voice_features_flow.png
  documentation/
    API_DOCUMENTATION.md
    CHEFVOICE_README.md
    TODO.md
  source_code/
    package.json
    pnpm-lock.yaml
    tsconfig.json
    vite.config.ts
    client/
      index.html
      public/
      src/
        App.tsx
        const.ts
        index.css
        main.tsx
    server/
      db.ts
      index.ts
      routers.ts
      storage.ts
      _core/
        context.ts
        cookies.ts
        dataApi.ts
        env.ts
        imageGeneration.ts
        index.ts
        llm.ts
        map.ts
        notification.ts
        oauth.ts
        sdk.ts
        systemRouter.ts
        trpc.ts
        vite.ts
        ... (+1 more files)
      data/
        irish-suppliers.ts
      routers/
        company-settings.test.ts
        company-settings.ts
        cooking-records.ts
        email-auth.ts
        extraction-voice.ts
        extraction.ts
        invite-codes.ts
        ocr.ts
        refrigeration.ts
        tts.ts
        validation.ts
        voice.ts
    shared/
      const.ts
      types.ts
```

```
_core/
  errors.ts
```

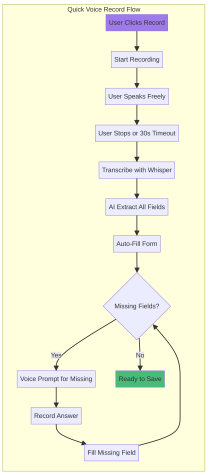
Core Diagrams

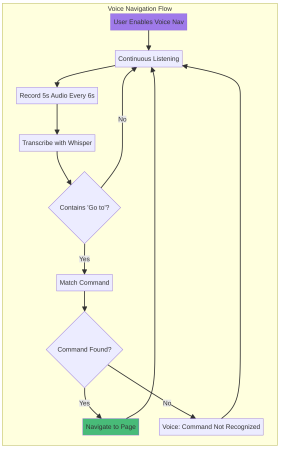
These diagrams reflect the current intended behaviour and system boundaries. They should be treated as design references during productionisation.

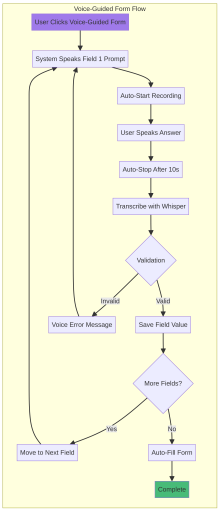
ChefVoice Architecture



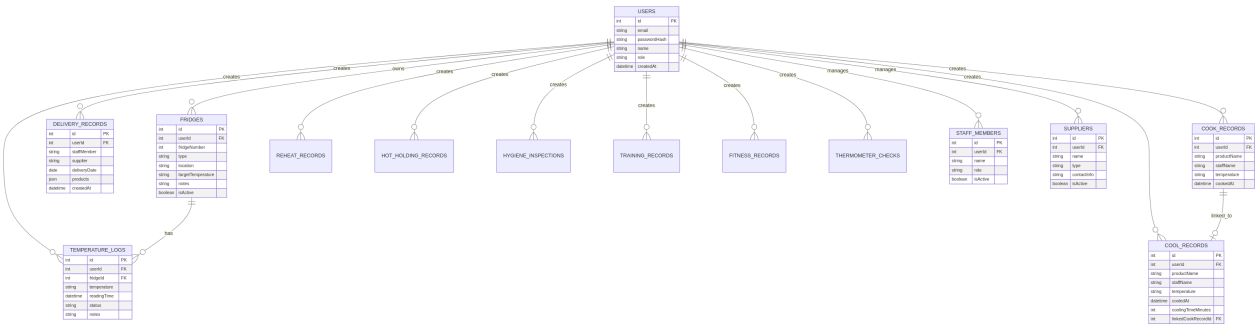
Voice Features Flow







Database Schema



Project README (from repo)

ChefVoice Beta Prototype - Complete Package

Version: 8384f73c
Date: January 2026
Package Contents: Source code, documentation, and architecture diagrams
---

■ Package Structure

```
chefvoice_package/
├── README.md                          # This file
├── diagrams/                          # Architecture diagrams
│   ├── chefvoice_architecture.png    # System architecture
│   ├── voice_features_flow.png       # Voice features workflow
│   ├── database_schema.png           # Database schema
│   ├── chefvoice_architecture.mmd    # Mermaid source
│   ├── voice_features_flow.mmd       # Mermaid source
│   └── database_schema.mmd           # Mermaid source
├── documentation/                     # Project documentation
│   ├── CHEFVOICE_README.md           # Comprehensive project README
│   ├── API_DOCUMENTATION.md          # Complete API reference
│   └── TODO.md                       # Development task list
└── source_code/                       # Full source code
    ├── client/                        # React frontend
    │   ├── src/
    │   │   ├── pages/                # Page components
    │   │   ├── components/           # UI components
    │   │   ├── hooks/                # Custom hooks
    │   │   ├── contexts/             # React contexts
    │   │   └── lib/                  # Utilities
    │   ├── public/                   # Static assets
    │   └── index.html                # Entry HTML
    ├── server/                        # Backend server
    │   ├── routes/                   # tRPC API routers
    │   ├── db/                       # Database schema
    │   └── index.ts                  # Server entry
    ├── shared/                        # Shared types
    └── package.json                  # Dependencies
```

■ Quick Start

```
unzip chefvoice_package.zip
cd chefvoice_package/source_code

pnpm install

DATABASE_URL=postgresql://user:password@host:port/database
JWT_SECRET=your-secret-key
OPENAI_API_KEY=sk-...
OAUTH_SERVER_URL=https://oauth.manus.im

pnpm db:push

pnpm dev
```

1. Extract the Package
2. Install Dependencies
3. Set Up Environment Variables
Create a `.env` file with:

4. Push Database Schema
5. Start Development Server

■ Documentation Guide

For Developers
1. **Start with**: `documentation/CHEFVOICE_README.md`
- Complete project overview
- Feature descriptions
- Architecture details
- Voice features documentation
2. **Then read**: `documentation/API_DOCUMENTATION.md`
- tRPC API reference
- All available endpoints
- Request/response formats
- Best practices
3. **Review**: `documentation/TODO.md`
- Current development status
- Completed features
- Pending tasks
For Architects
1. **View**: `diagrams/chefvoice_architecture.png`
- System components
- Data flow
- External services
2. **Study**: `diagrams/voice_features_flow.png`
- Voice-guided form workflow
- Voice navigation flow
- Quick voice record process
3. **Examine**: `diagrams/database_schema.png`
- Entity relationships
- Table structures
- Foreign key constraints

■ Key Features

Voice-Powered Operations
• **Voice-Guided Forms**: Sequential field-by-field prompts
• **Voice Navigation**: Hands-free page navigation
• **Quick Voice Record**: Natural language form filling
• **Voice Alerts**: Audio feedback for all actions
HACCP Compliance
• Complete FSAI forms SC1-SC7
• Temperature monitoring with alerts
• Delivery tracking with OCR
• Comprehensive reporting
Modern Tech Stack
• React 19 + TypeScript
• tRPC for type-safe APIs

• PostgreSQL database
• OpenAI Whisper + GPT-4o
• Tailwind CSS + shadcn/ui

■ Important Files

Frontend Entry Points
• `client/src/App.tsx` - Main application router
• `client/src/pages/Dashboard.tsx` - Dashboard home
• `client/src/pages/Delivery.tsx` - Delivery tracking (with voice-guided form)
• `client/src/pages/Settings.tsx` - Voice navigation settings
Voice Features
• `client/src/hooks/useVoiceGuidedForm.tsx` - Voice-guided form hook
• `client/src/hooks/useVoiceNavigation.tsx` - Voice navigation hook
• `client/src/hooks/useVoiceRecording.tsx` - Voice recording utilities
• `client/src/contexts/VoiceSettingsContext.tsx` - Voice settings management
Backend API
• `server/routes/voice.ts` - Voice transcription
• `server/routes/extraction.ts` - AI field extraction
• `server/routes/ocr.ts` - Invoice OCR
• `server/routes/refrigeration.ts` - Temperature logging
• `server/routes/records.ts` - HACCP records
Database
• `server/db/schema.ts` - Drizzle ORM schema
• All tables for users, fridges, logs, records

■ Configuration Files

• `package.json` - Dependencies and scripts
• `tsconfig.json` - TypeScript configuration
• `vite.config.ts` - Vite build configuration
• `tailwind.config.ts` - Tailwind CSS customization
• `client/src/index.css` - Global styles and theme

■ Customization

Branding
• Update `VITE_APP_TITLE` in environment variables
• Replace `VITE_APP_LOGO` with your logo URL
• Modify color palette in `client/src/index.css`
Voice Settings
• Adjust default speed/pitch/volume in `VoiceSettingsContext.tsx`
• Add new voice commands in `useVoiceNavigation.tsx`
• Customize prompts in `useVoiceGuidedForm.tsx`
Forms
• Add new pages in `client/src/pages/`
• Create tRPC routers in `server/routes/`

• Define database tables in `server/db/schema.ts`

■ Troubleshooting

```
# Clear cache and reinstall
rm -rf node_modules pnpm-lock.yaml
pnpm install

# Reset database schema
pnpm db:push --force
```

Build Errors
Database Issues
Voice Not Working
• Check microphone permissions in browser
• Verify OPENAI_API_KEY is set
• Test with Chrome/Edge (best compatibility)

■ Support

• **Documentation**: See `documentation/` folder
• **Issues**: Check `documentation/TODO.md` for known issues
• **Help**: https://help.manus.im

■ License

Proprietary beta prototype. All rights reserved.

■ Credits

Built with Manus platform and OpenAI APIs.
Version: 8384f73c
Last Updated: January 2026

ChefVoice Product/Technical README

ChefVoice Beta Prototype

Voice-Powered Kitchen Operations Management System
ChefVoice is a comprehensive HACCP compliance and food safety management platform designed specifically for restaurant kitchens. It enables completely hands-free operation through advanced voice recognition, AI-powered form filling, and voice navigation.

■ Project Overview

ChefVoice helps restaurants streamline temperature monitoring, delivery tracking, and compliance documentation with hands-free voice commands. The system is built for busy kitchen environments where staff need to log critical food safety data without touching devices with contaminated hands.
Key Features
• **Voice-Guided Form Filling**: Sequential field-by-field voice prompts that guide users through complete forms
• **Voice Navigation**: Hands-free navigation between all dashboard features using "Go to [page]" commands
• **Quick Voice Recording**: Speak naturally and AI extracts all form fields automatically
• **OCR Photo Capture**: Take pictures of invoices and automatically extract delivery data
• **Real-Time Temperature Monitoring**: Track refrigeration units with compliance alerts
• **HACCP Compliance**: Complete FSAI forms SC1-SC7 for Irish food safety regulations
• **Voice Alerts**: Audio feedback for all critical actions and temperature violations
• **Multi-Input Methods**: Voice, text, and photo capture for maximum flexibility
---

■■ Architecture

```
chefvoice_beta/
├── client/                # Frontend React application
│   ├── src/
│   │   ├── pages/         # Page components
│   │   │   ├── Landing.tsx
│   │   │   ├── Dashboard.tsx
│   │   │   ├── Delivery.tsx
│   │   │   ├── Refrigeration.tsx
│   │   │   ├── CookCoolReheat.tsx
│   │   │   ├── HotHolding.tsx
│   │   │   ├── HygieneInspection.tsx
│   │   │   ├── HygieneTraining.tsx
│   │   │   ├── FitnessToWork.tsx
│   │   │   ├── ThermometerCheck.tsx
│   │   │   ├── Reports.tsx
│   │   │   ├── ComplianceDashboard.tsx
│   │   │   └── Settings.tsx
│   │   ├── components/    # Reusable UI components
│   │   ├── hooks/         # Custom React hooks
│   │   │   ├── useVoiceGuidedForm.tsx
│   │   │   ├── useVoiceNavigation.tsx
│   │   │   ├── useVoiceRecording.tsx
│   │   │   └── useVoiceConfirmation.tsx
│   │   ├── contexts/      # React contexts
│   │   │   └── VoiceSettingsContext.tsx
│   │   └── lib/           # Utilities
├── server/                # Backend tRPC server
│   ├── routes/           # API routers
│   │   ├── voice.ts
```

```
│   │   ├── extraction.ts
│   │   ├── ocr.ts
│   │   └── refrigeration.ts
│   └── db/               # Database schema
└── shared/               # Shared types and constants
```

Technology Stack

Frontend:

• React 19 with TypeScript
• Wouter for routing
• TailwindCSS 4 for styling
• shadcn/ui component library
• Web Speech API for voice synthesis
• OpenAI Whisper for transcription

Backend:

• Node.js with Express
• tRPC for type-safe API
• Drizzle ORM
• PostgreSQL database

External Services:

• OpenAI Whisper API (via Manus Forge)
• OpenAI GPT-4o for AI extraction
• Manus hosting platform

Project Structure

■ Voice Features

1. Voice-Guided Form Filling

How it works:

1. User clicks "Voice-Guided Form" button
2. System speaks first field prompt: "Please say your name"
3. Automatically starts recording for 10 seconds
4. Transcribes response using OpenAI Whisper
5. Validates the response
6. If valid, moves to next field; if invalid, re-prompts
7. Continues until all fields are complete
8. Auto-fills the entire form with collected data

Benefits:

• Structured data collection
• Field-by-field validation
• Clear guidance for users
• Reduces errors from free-form speech

Implementation:

• Hook: `useVoiceGuidedForm.tsx`
• Used in: Delivery page (6 fields)
• Can be extended to any form

2. Voice Navigation Commands

How it works:

1. User enables voice navigation in Settings
2. System listens continuously every 6 seconds
3. Records 5-second audio clips
4. Transcribes and checks for "Go to [page]" pattern
5. Matches keywords to available pages

6. Navigates automatically with voice confirmation
Available Commands:
• "Go to home" / "Go to dashboard"
• "Go to refrigeration" / "Go to fridge"
• "Go to delivery"
• "Go to cook" / "Go to cooking"
• "Go to hot holding"
• "Go to hygiene inspection"
• "Go to training"
• "Go to fitness"
• "Go to thermometer"
• "Go to reports"
• "Go to compliance"
• "Go to settings"
Benefits:
• Completely hands-free navigation
• No need to touch screen between tasks
• Ideal for contaminated hands in kitchen
Implementation:
• Hook: `useVoiceNavigation.tsx`
• Enabled in: Settings page
• Works across all pages
3. Quick Voice Recording
How it works:
1. User clicks "Quick Voice Record" button
2. Speaks naturally about the delivery/task
3. System records until user stops or 30s timeout
4. Transcribes entire recording
5. AI extracts all relevant fields using GPT-4o
6. Auto-fills form with extracted data
7. Voice prompts for any missing mandatory fields
Benefits:
• Natural conversation style
• Fastest input method
• AI handles field extraction
• Automatic missing field detection
Implementation:
• Used in: Delivery, Refrigeration, Cook/Cool/Reheat, Hot Holding
• AI extraction via `extraction.ts` router
4. Voice Alerts
How it works:
• All toast notifications are accompanied by voice feedback
• Temperature violations trigger voice warnings
• Success confirmations are spoken aloud
• Error messages are read to user
Customization:
• Speed: 0.5x - 2.0x
• Pitch: 0 - 2
• Volume: 0% - 100%
• Settings saved in localStorage
Implementation:

• Context: `VoiceSettingsContext.tsx`
• Used across all pages

■ FSAI Compliance Forms

ChefVoice implements all required FSAI (Food Safety Authority of Ireland) forms:
SC1 - Delivery Tracking
• Supplier name and delivery date
• Product details with batch numbers
• Temperature checks (Frozen/Chilled/Ambient)
• Quality assessment
• Staff member logging
SC2 - Refrigeration Monitoring
• Multiple fridge/freezer tracking
• Temperature logging with timestamps
• Compliance status (OK/Warning/Critical)
• 60-minute reminder alerts
• Historical temperature logs
SC3 - Cook, Cool & Reheat
• Cooking temperatures (≥75°C)
• Cooling times and temperatures
• Reheating temperatures (≥75°C)
• Linked cook-cool records for traceability
SC4 - Hot Holding
• Food item and time into hot hold
• Core temperature checks (≥63°C)
• 90-minute check intervals
• Staff member tracking
SC5 - Hygiene Inspection Checklist
• Weekly/monthly inspection schedules
• 12 checkpoint categories
• Pass/fail status per checkpoint
• Inspector name and notes
SC6 - Hygiene Training Records
• Staff member training dates
• Training topics and duration
• Trainer name
• Refresher due dates
SC7 - Fitness to Work Assessment
• Daily health checks
• Symptom tracking
• Fit/unfit determination
• Manager approval
Additional Forms
• **Thermometer Calibration**: Ice bath and boiling water checks
• **Staff Registry**: Registered staff names for better voice recognition
• **Supplier Registry**: Known suppliers for spell-checking

■ Reports & Compliance

Reports Page Features
• **Activity Timeline**: Chronological view of all records
• **Filtering**: By date range and record type
• **Summary Cards**: Quick stats for each category
• **FSAI Export**: Text format for official submissions
• **PDF Export**: Professional branded reports with tables
• **Color-Coded Status**: Visual compliance indicators
Compliance Dashboard
• **Overdue Items**: Training refreshers, thermometer checks, inspections
• **Upcoming Alerts**: Items due in next 7 days
• **Staff Fitness**: Current health status
• **Action Buttons**: Quick links to complete tasks

■ Getting Started

```
cd /home/ubuntu/chefvoice_beta

pnpm install

# Already configured in Manus:
# - OPENAI_API_KEY
# - JWT_SECRET
# - DATABASE_URL
# - OAUTH_SERVER_URL

pnpm db:push

pnpm dev
```

Prerequisites
• Node.js 22+
• PostgreSQL database
• OpenAI API key
Installation
1. **Clone the repository**
2. **Install dependencies**
3. **Set up environment variables**
4. **Push database schema**
5. **Start development server**
6. **Access the application**
• Development: https://3000-[sandbox-id].manusvm.computer
• Production: https://[your-domain].manus.space

■ Design System

Color Palette
• **Primary**: Purple (#9f7aea) - Voice features
• **Success**: Green (#48bb78) - Compliant temperatures
• **Warning**: Yellow (#ecc94b) - Near-limit temperatures
• **Danger**: Red (#f56565) - Non-compliant temperatures
• **Info**: Blue (#4299e1) - Refrigeration, general info

Typography
• **Headings**: Bold, 2xl-4xl sizes
• **Body**: Regular, base size
• **Labels**: Medium weight, sm size
Voice UI Patterns
• **Recording State**: Red pulsing button with timer
• **Voice Active**: Purple background with microphone icon
• **Voice Prompts**: Purple info cards with current field
• **Voice Navigation**: Green indicator when listening

■ Configuration

Voice Settings
Accessible in Settings page:
• **Speed**: Adjust speech rate (0.5x - 2.0x)
• **Pitch**: Change voice tone (0 - 2)
• **Volume**: Control loudness (0% - 100%)
• **Test Button**: Preview settings
Voice Navigation
Enable/disable in Settings:
• Toggle button to activate continuous listening
• Visual indicator when active
• List of available commands
• Works across all pages
Company Settings
Register staff and suppliers:
• **Staff Members**: Name, role, active status
• **Suppliers**: Name, type, contact info
• Improves voice recognition accuracy
• Better AI extraction from voice/OCR

■ Mobile Optimization

ChefVoice is fully optimized for mobile devices:
• **Touch-Friendly**: Minimum 44px tap targets
• **Responsive Layout**: Adapts to all screen sizes
• **Hamburger Menu**: Collapsible navigation on mobile
• **Large Buttons**: Easy to tap with gloves
• **Voice-First**: Minimal touch required

■ Security & Authentication

• **User Authentication**: Email/password with JWT tokens
• **Multi-Tenant**: Data isolation per restaurant
• **Role-Based Access**: Admin and staff roles
• **Secure API**: tRPC with type safety
• **Environment Variables**: Secrets managed via Manus

■ Future Enhancements

Planned Features
1. **Multi-Language Support**: Irish, Polish, Spanish for international staff
2. **Voice Command Shortcuts**: "Save record", "Take photo", "Show reports"
3. **Offline Mode**: Continue logging without internet
4. **Mobile App**: Native iOS/Android apps
5. **Sensor Integration**: Bluetooth temperature probes
6. **Automated Scheduling**: Recurring tasks and reminders
7. **Advanced Analytics**: Trend analysis and predictions
8. **Email Reports**: Automated daily/weekly summaries
Voice Improvements
1. **Custom Wake Words**: "Hey ChefVoice" instead of button press
2. **Accent Training**: Better recognition for diverse accents
3. **Noise Filtering**: Improved kitchen noise handling
4. **Voice Personas**: Different voices for different alert types
5. **Conversation Mode**: Multi-turn dialogues for complex tasks
---

■ Known Issues & Limitations

Voice Recognition
• Requires microphone permissions
• May struggle with heavy background noise
• Accents may affect accuracy
• Internet required for transcription
Browser Compatibility
• Best on Chrome/Edge (Web Speech API support)
• Safari has limited speech synthesis
• Firefox requires additional permissions
Performance
• Large record sets may slow Reports page
• Voice navigation uses continuous recording (battery impact)
• OCR requires good photo quality

■ Support & Contact

For technical support, feature requests, or bug reports:
• **Website**: https://help.manus.im
• **Documentation**: See inline comments in code
• **API Docs**: tRPC router definitions in `server/routes/`

■ License

■ Acknowledgments

• **OpenAI**: Whisper and GPT-4o APIs
• **Manus**: Hosting and Forge API infrastructure
• **FSAI**: Food safety guidelines and form templates
• **shadcn/ui**: Beautiful component library
• **Tailwind CSS**: Utility-first styling

■ Version History

v8384f73c (Current)
• ■ Added back-to-dashboard buttons on all form pages
• ■ Consistent navigation across entire app
• ■ One-tap return to dashboard from any page
v7b20a92a
• ■ Implemented voice-guided form filling
• ■ Added voice navigation commands
• ■ Sequential field prompts with validation
• ■ Continuous listening for navigation
Previous Versions
• Voice alerts across all features
• OCR photo capture for deliveries
• Complete FSAI SC1-SC7 forms
• Compliance dashboard
• PDF export with branding
• Staff and supplier registry
• Mobile optimization

Built with ❤■ for restaurant kitchens everywhere

API Documentation (excerpt)

ChefVoice API Documentation

Overview

ChefVoice uses **tRPC** for type-safe API communication between the frontend and backend. All API routes are defined in the `server/routes/` directory and automatically provide TypeScript types to the client.

Base Configuration

Endpoint: Configured via environment variables
Authentication: JWT tokens in HTTP headers
Type Safety: Full TypeScript inference via tRPC

API Routers

```
{
  audioBase64: string;  // Base64-encoded audio data
  mimeType: string;     // Audio format (e.g., "audio/webm")
}

{
  text: string;         // Transcribed text
  language?: string;    // Detected language
  duration?: number;    // Audio duration in seconds
}

const result = await trpc.voice.transcribe.mutateAsync({
  audioBase64: base64Audio,
  mimeType: "audio/webm"
});
console.log(result.text);

{
  transcription: string;     // Raw transcription text
  formType: string;          // Type of form (e.g., "delivery", "temperature")
  registeredStaff?: string[];   // Known staff names
  registeredSuppliers?: string[]; // Known supplier names
}

{
  staffMember?: string;
  supplier?: string;
  products?: Array<{
    name: string;
    quantity: string;
    temperature: string;
    category: "Frozen" | "Chilled" | "Ambient";
    batchNumber: string;
    quality: string;
  }>;
  deliveryDate?: string;
  missingFields?: string[];
}

const extracted = await trpc.extraction.extractFormFields.mutateAsync({
  transcription: "Delivery from Fresh Foods, 10 kg salmon at -18 degrees",
  formType: "delivery",
  registeredSuppliers: ["Fresh Foods Ltd", "Ocean Catch"]
```

```
});

{
  imageBase64: string;       // Base64-encoded image
  registeredSuppliers?: string[];
}

{
  supplier: string;
  deliveryDate: string;
  products: Array<{
    name: string;
    quantity: string;
    temperature?: string;
    storageCategory: "Frozen" | "Chilled" | "Ambient";
    batchNumber?: string;
  }>;
  confidence: number;        // 0-1 confidence score
}

const ocrResult = await trpc.ocr.analyzeInvoice.mutateAsync({
  imageBase64: base64Image,
  registeredSuppliers: supplierList
});

{
  fridgeNumber: number;
  type: string;              // "Fridge", "Freezer", "Blast Chiller"
  location?: string;
  targetTemperature?: string;
  notes?: string;
}

{
  id: number;
  userId: number;
  fridgeNumber: number;
  type: string;
  location: string | null;
  targetTemperature: string | null;
  notes: string | null;
  isActive: boolean;
  createdAt: Date;
  updatedAt: Date;
}

Array<Fridge>

{
  fridgeId: number;
  temperature: string;
  notes?: string;
}

{
  id: number;
  userId: number;
  fridgeId: number;
  temperature: string;
  readingTime: Date;
  status: "ok" | "warning" | "critical";
  notes: string | null;
  createdAt: Date;
}

Array<TemperatureLog>

{
  fridgeId: number;
}
```

```
{
  success: boolean;
}

Input: {
  productName: string;
  staffName: string;
  temperatureAfterCooking: string;
  cookedAt: Date;
}
Output: CookRecord

Input: None
Output: Array<CookRecord>

Input: {
  productName: string;
  staffName: string;
  temperatureAfterCooling: string;
  cooledAt: Date;
  coolingTimeMinutes?: number;
  linkedCookRecordId?: string;
}
Output: CoolRecord

Input: None
Output: Array<CoolRecord>

Input: {
  productName: string;
  staffName: string;
  temperatureAfterReheating: string;
  reheatedAt: Date;
}
Output: ReheatRecord

Input: None
Output: Array<ReheatRecord>

Input: {
  foodItem: string;
  timeIntoHotHold: string;
  coreTemperature: string;
  checkedBy: string;
  comments?: string;
}
Output: HotHoldingRecord

Input: None
Output: Array<HotHoldingRecord>

Input: {
  inspectionDate: Date;
  inspectorName: string;
  checkpoints: Array<{
    category: string;
    status: "pass" | "fail";
    notes?: string;
  }>;
}
Output: HygieneInspectionRecord

Input: {
  staffMember: string;
  trainingDate: Date;
  topic: string;
  duration: number;
  trainer: string;
  refresherDue?: Date;
}
```

```
Output: TrainingRecord

Input: {
  staffMember: string;
  assessmentDate: Date;
  symptoms: string[];
  fitToWork: boolean;
  managerName: string;
}
Output: FitnessRecord

Input: {
  checkDate: Date;
  thermometerNumber: string;
  iceWaterReading: string;
  boilingWaterReading: string;
  checkedBy: string;
  calibrationStatus: "pass" | "fail";
  nextCheckDue: Date;
}
Output: ThermometerCheckRecord

{
  name: string;
  role: string;
}

{
  id: number;
  userId: number;
  name: string;
  role: string;
  isActive: boolean;
  createdAt: Date;
}

Array<StaffMember>

{
  staffId: number;
}

{
  success: boolean;
}

{
  name: string;
  type: string;
  contactInfo?: string;
}

{
  id: number;
  userId: number;
  name: string;
  type: string;
  contactInfo: string | null;
  isActive: boolean;
  createdAt: Date;
}

Array<Supplier>

{
  supplierId: number;
}

{
  success: boolean;
}
```

1. Voice Router (`voice.ts`)
Handles all voice transcription operations.
`voice.transcribe`
Transcribes audio to text using OpenAI Whisper API.
Input:
Output:
Usage:
---
2. Extraction Router (`extraction.ts`)
AI-powered field extraction from transcriptions.
`extraction.extractFormFields`
Extracts structured form data from natural language transcription.
Input:
Output:
Usage:
---
3. OCR Router (`ocr.ts`)
Optical character recognition for invoice scanning.
`ocr.analyzeInvoice`
Analyzes invoice photos and extracts delivery data.
Input:
Output:
Usage:
---
4. Refrigeration Router (`refrigeration.ts`)
Manages refrigeration units and temperature logs.
`refrigeration.createFridge`
Creates a new refrigeration unit.
Input:
Output:
`refrigeration.getAllFridges`
Retrieves all refrigeration units for the current user.
Input: None
Output:
`refrigeration.logTemperature`
Logs a temperature reading for a fridge.
Input:
Output:
`refrigeration.getAllLogs`
Retrieves all temperature logs for the current user.
Input: None
Output:
`refrigeration.deleteFridge`
Soft-deletes a refrigeration unit.
Input:
Output:
---
5. Records Router (`records.ts`)
Manages all HACCP compliance records.
Cook Records
`records.createCookRecord`
`records.getAllCookRecords`

Cool Records
`records.createCoolRecord`
`records.getAllCoolRecords`
Reheat Records
`records.createReheatRecord`
`records.getAllReheatRecords`
Hot Holding Records
`records.createHotHoldingRecord`
`records.getAllHotHoldingRecords`
Hygiene Inspection Records
`records.createHygieneInspection`
Training Records
`records.createTrainingRecord`
Fitness to Work Records
`records.createFitnessRecord`
Thermometer Check Records
`records.createThermometerCheck`
---
6. Staff Router (`staff.ts`)
Manages registered staff members.
`staff.createStaff`
Input:
Output:
`staff.getAllStaff`
Input: None
Output:
`staff.deleteStaff`
Input:
Output:
---
7. Supplier Router (`supplier.ts`)
Manages registered suppliers.
`supplier.createSupplier`
Input:
Output:
`supplier.getAllSuppliers`
Input: None
Output:
`supplier.deleteSupplier`
Input:
Output:

Error Handling

```
try {
  const result = await trpc.someRoute.mutateAsync(input);
} catch (error) {
  if (error instanceof TRPCError) {
    console.error(error.code, error.message);
    // Handle specific error codes:
    // - UNAUTHORIZED: User not authenticated
    // - BAD_REQUEST: Invalid input
```

```
    // - INTERNAL_SERVER_ERROR: Server error
  }
}
```

All tRPC routes use standardized error handling:

Authentication

All API routes (except public ones) require authentication:
1. User logs in via `/api/auth/login`
2. JWT token is stored in HTTP-only cookie
3. Token is automatically sent with all tRPC requests
4. Server validates token and attaches user to context
Protected Routes:
• All `refrigeration.*` routes
• All `records.*` routes
• All `staff.*` and `supplier.*` routes
Public Routes:
• `voice.transcribe` (uses API key authentication)
• `extraction.extractFormFields`
• `ocr.analyzeInvoice`

Rate Limiting

API routes have the following rate limits:
• **Voice Transcription**: 100 requests/minute per user
• **AI Extraction**: 50 requests/minute per user
• **OCR Analysis**: 20 requests/minute per user
• **Database Operations**: 1000 requests/minute per user
Exceeding rate limits returns HTTP 429 with retry-after header.

Best Practices

```
// Query (GET)
const { data, isLoading } = trpc.refrigeration.getAllFridges.useQuery();

// Mutation (POST/PUT/DELETE)
const createMutation = trpc.refrigeration.createFridge.useMutation({
  onSuccess: () => {
    // Invalidate cache
    trpcUtils.refrigeration.getAllFridges.invalidate();
  }
});

if (isLoading) return <Spinner />;
if (error) return <Error message={error.message} />;
return <DataDisplay data={data} />;

const deleteMutation = trpc.refrigeration.deleteFridge.useMutation({
  onMutate: async (variables) => {
    // Cancel outgoing queries
    await trpcUtils.refrigeration.getAllFridges.cancel();

    // Snapshot current value
    const previous = trpcUtils.refrigeration.getAllFridges.getData();
```

```
    // Optimistically update
    trpcUtils.refrigeration.getAllFridges.setData(undefined, (old) =>
      old?.filter(f => f.id !== variables.fridgeId)
    );

    return { previous };
  },
  onError: (err, variables, context) => {
    // Rollback on error
    trpcUtils.refrigeration.getAllFridges.setData(undefined, context?.previous);
  }
});

// These will be batched into a single HTTP request
const [fridges, logs, staff] = await Promise.all([
  trpc.refrigeration.getAllFridges.query(),
  trpc.refrigeration.getAllLogs.query(),
  trpc.staff.getAllStaff.query()
]);
```

1. Use React Query Hooks

tRPC provides React Query hooks for optimal caching:

2. Handle Loading States

3. Optimistic Updates

4. Batch Requests

tRPC automatically batches requests made within the same tick:

Testing

```
import { createCaller } from '../server/trpc';

describe('Refrigeration Router', () => {
  it('creates a fridge', async () => {
    const caller = createCaller({ user: mockUser });
    const result = await caller.refrigeration.createFridge({
      fridgeNumber: 1,
      type: 'Fridge',
      targetTemperature: '0-5°C'
    });
    expect(result.id).toBeDefined();
  });
});

import { render, screen } from '@testing-library/react';
import { trpc } from '@/lib/trpc';

test('displays fridges', async () => {
  render(<RefrigerationPage />);
  await screen.findByText('Fridge #1');
  expect(screen.getByText('0-5°C')).toBeInTheDocument();
});
```

Unit Tests

Integration Tests

Deployment

```
DATABASE_URL=postgresql://...
JWT_SECRET=your-secret-key
OPENAI_API_KEY=sk-...
VITE_FRONTEND_FORGE_API_KEY=...
VITE_FRONTEND_FORGE_API_URL=...
```

```
# Build frontend
cd client && pnpm build

# Build backend
cd server && pnpm build

# Start production server
pnpm start
```

Environment Variables
Required for production:
Build Process

Support

For API-related questions:
• Check tRPC router definitions in `server/routes/`
• Review type definitions in `shared/types/`
• Test endpoints using tRPC DevTools
• Contact support at https://help.manus.im

TODO / Backlog (prioritised excerpt)

The following excerpt highlights the highest priority items and productionisation tasks identified in the current backlog. Full TODO remains in the repository.

ChefVoice Beta Prototype - TODO

Core Features

• [x] Voice recording interface with start/stop controls
• [x] Real-time speech-to-text transcription using Web Speech API
• [x] Food Delivery Record (SC1) form with fields:
- [x] Date of Delivery (auto-populated from voice or manual)
- [x] Supplier Name
- [x] Product Description
- [x] Quantity
- [x] Temperature (if applicable)
- [x] Quality Assessment (voice-driven)
• [x] Form submission and data storage (local storage for MVP)
• [x] Display submitted records in a list/table view
• [x] Clear UI with visual feedback for recording status
• [x] Responsive design for mobile/tablet use in kitchen environment

UI/UX

• [x] Professional kitchen-focused design
• [x] Large, easy-to-tap buttons for kitchen use
• [x] Visual indicators for recording status (red dot, timer)
• [x] Success/error notifications
• [x] Accessibility features (high contrast, readable fonts)

Testing & Deployment

• [x] Fix speech recognition error and improve error handling
• [x] Add manual text input fallback for testing without speech recognition
• [x] Integrate OpenAI Whisper API for professional speech-to-text
• [x] Add AI-powered automatic form field extraction from transcriptions
• [x] Fix AI form field extraction - returning blank fields
• [x] Add staff member field to track who logged each delivery
• [x] Add AI validation for missing form fields with user prompts
• [x] Add temperature validation with safe range alerts
• [x] Fix staff member name extraction from voice transcriptions
• [x] Add supplier name spell-checking against Irish food suppliers database
• [x] Improve staff member name recognition with flexible patterns
• [x] Add support for multiple products in single delivery
• [x] Implement temperature category separation (Frozen, Chilled, Ambient)
• [x] Add batch number field to product tracking
• [x] Implement AI voice prompts for missing form information
• [x] Fix supplier name spell-checking not applying correctly
• [x] Fix AI voice prompts not triggering for missing fields
• [x] Fix validation to alarm on missing mandatory fields before saving

• [x] Add "Hey Chef!" wake word activation for automatic recording
• [x] Add automatic stop recording when speech ends (silence detection)
• [x] Implement continuous background listening for "Hey Chef!" wake word
• [x] Fix: Background listening should be ON by default on page load
• [x] Fix: "Hey Chef!" wake word detection and auto start/stop not working
• [x] Debug: Background listening not detecting "Hey Chef!" wake word properly (fixed async/await flow)
• [x] Implement stop phrase detection ("That's it") to end recording
• [x] Add noise filtering to ignore background kitchen sounds
• [x] Improve recording focus to only capture relevant delivery information
• [x] Fix: Speech synthesis getting stuck during validation prompts
• [x] Fix: Recording failing with "Failed to start recording" error
• [x] Add OCR photo capture feature using OpenAI Vision API
• [x] Extract product names, dates, batch codes from photos
• [x] Auto-populate form with OCR-extracted data
• [x] Fix OCR photo analysis - Vision API call failing
• [x] Fix OCR model access - gpt-4-turbo not available, use gpt-4o instead
• [x] Resolve OpenAI API quota issue (Error 429) - user added funds and fixed OCR router to use environment variable
• [x] Fix: OPENAI_API_KEY environment variable not being properly injected - registered via webdev_request_secrets
• [x] Implement Reports page with filtering and compliance status display
• [x] Add FSAI form generation and text export functionality
• [x] Add Reports tab and card to dashboard navigation
• [x] Implement database schema for cooking, cooling, and reheating records
• [x] Create backend tRPC API endpoints for record CRUD operations
• [ ] Test voice recognition with various accents and kitchen noise
• [ ] Verify form data capture accuracy
• [ ] Test on mobile devices (iOS/Android browsers)
• [ ] Deploy to public URL for external user access (ready for checkpoint)
• [ ] Create user guide/instructions for beta testers

Future Enhancements (Post-MVP)

• [ ] Backend integration for data persistence
• [ ] OCR for invoice scanning
• [ ] Temperature sensor integration
• [ ] Risk assessment model
• [ ] User authentication and multi-user support
• [x] Fix: Background listening automatically starts on page load without user permission
• [x] Fix: Transcription errors from automatic empty audio recording
• [x] Fix: OPENAI_API_KEY environment variable not being passed to frontend for Whisper API (uses Forge API instead)
• [x] Make batch number mandatory field - prevent saving without batch numbers for all products
• [x] Add voice prompt asking for batch number if missing
• [x] Improve OCR AI prompt to understand supplier context and product categories
• [x] Teach AI that fish/seafood is always chilled or frozen, not ambient
• [x] Add product-specific storage rules (dairy, meat, produce, etc.) to OCR analysis
• [x] Fix: OCR still categorizing fish products as Ambient instead of Chilled/Frozen
• [x] Fix: OCR not extracting batch code from invoice number/reference fields
• [x] Add logic to use invoice number/reference as batch code when batch code not explicitly labeled
• [x] Add post-processing logic to force-correct fish products to Chilled/Frozen if OCR returns Ambient

• [x] Fix: Frontend was ignoring storageCategory from OCR and using temperature instead
• [x] Fix: Frontend now uses storageCategory from backend for correct product categorization
• [x] Add voice-first missing field prompts - automatically open mic and ask for missing data via voice
• [x] Allow text input as fallback when voice input is not available
• [x] Parse voice responses to extract temperature values and other missing fields
• [x] Bug: OCR analysis fails and causes dev server to crash when analyzing pictures (fixed with audio stream initialization)
• [x] Bug: Voice-first missing field prompts don't open mic for staff member name field (fixed async/await handling)
• [x] Bug: Voice-first missing field prompts don't transcribe after OCR (fixed with audio stream initialization)
• [x] Bug: Batch number extraction from voice responses not working (added batch number extraction logic)
• [x] Fix: TypeScript errors in Web Speech API types (added type assertions)
• [x] Fix: Audio recording failing with empty blobs (create fresh MediaRecorder for each session)
• [x] All three input methods (voice, text, OCR) now have working voice-first missing field prompts
• [ ] Implement category-grouped temperature prompts (group by Frozen/Chilled/Ambient)
• [ ] Parse voice response to extract temperatures for each category
• [ ] Apply category temperatures to all products in that category
• [ ] Add yellow warning indicator when temperature is close to safe limits
• [ ] Add red alert indicator when temperature is outside safe limits
• [ ] Emit voice alert when temperature is out of range
• [ ] Ask user to correct temperature via voice prompt
• [ ] Remind user to update temperature again in 60 minutes
• [ ] Implement automatic 60-minute reminder timer after saving temperature
• [ ] Auto-trigger alert after 60 minutes with sound notification
• [ ] Remind user to log temperature again after 60 minutes
• [x] Create public SaaS landing page with hero section
• [x] Add feature highlights section (voice monitoring, delivery tracking, alerts)
• [x] Add pricing section on landing page
• [x] Add sign-up/login buttons on landing page
• [x] Implement user authentication system (login/signup)
• [x] Create restaurant dashboard with persistent navigation
• [x] Add Home, Refrigeration, Delivery, Settings tabs to dashboard
• [x] Integrate refrigeration page into dashboard
• [x] Create delivery form template and structure
• [x] Add restaurant settings page (manage users, restaurant info)
• [ ] Implement multi-tenant data isolation
• [x] Test authentication and navigation between all pages
• [x] Implement voice recording for delivery form
• [x] Add voice transcription parsing for delivery data extraction
• [x] Test voice recording and auto-fill functionality for delivery form
• [x] Rebuild delivery form with complete fields (staff, date, supplier, condition type, quality assessment)
• [ ] Implement smart conversational voice parsing (supplier + products only)
• [ ] Create form to collect missing mandatory fields after voice input
• [x] Rebuild delivery form with AI extraction router
• [x] Add mandatory field validation (staff, temperature, batch number)
• [x] Implement temperature category tabs (Frozen/Chilled/Ambient)
• [ ] Implement AI voice prompts for missing mandatory fields
• [ ] Parse voice responses to auto-fill missing fields
• [ ] Fix OCR photo analysis to properly extract invoice data
• [ ] Test complete delivery workflow with voice and OCR
• [x] Fix Reports page to display linked cooking and cooling records together in unified activity timeline

- [x] Add color-coded activity cards (blue for linked, orange for cook-only, purple for cool-only)
- [x] Display both cooking and cooling data in single activity entry with full HACCP traceability
- [x] Fix cooling and reheating records not saving to localStorage
- [x] Add localStorage.setItem() calls to saveCoolRecord and saveReheatRecord functions

Hot Holding Feature (FSAI SC4)

- [x] Create HotHolding page component with FSAI SC4 form
- [x] Add voice recording and transcription for hot holding records
- [x] Implement form fields: Date, Food, Time Into Hot Hold, Core Temp, Checked By, Comments
- [x] Add temperature validation (must be >63°C for compliance)
- [x] Add hot holding records to localStorage for persistence
- [x] Integrate Hot Holding tab into Dashboard navigation
- [x] Display hot holding records in Reports page
- [x] Add FSAI SC4 export functionality to Reports
- [x] Update hot holding check interval from 2 hours to 90 minutes in documentation

FSAI Additional Forms

- [x] Research FSAI SC5 requirements (Hygiene Inspection Checklist)
- [x] Research FSAI SC6 requirements (Hygiene Training Records)
- [x] Research FSAI SC7 requirements (Fitness to Work Assessment)
- [x] Research thermometer check/calibration standards
- [x] Create SC5 Hygiene Inspection page component
- [x] Create SC6 Hygiene Training page component
- [x] Create SC7 Fitness to Work page component
- [x] Create Thermometer Check page component
- [x] Add SC5, SC6, SC7, Thermometer tabs to Dashboard navigation
- [x] Add new form cards to Dashboard home page
- [x] Add routes for all new forms to App.tsx
- [x] Add SC5, SC6, SC7, Thermometer interfaces to Reports page
- [x] Load new record types from localStorage in Reports
- [x] Add summary cards for new record types
- [x] Create display sections for SC5 Hygiene Inspection records
- [x] Create display sections for SC6 Training records
- [x] Create display sections for SC7 Fitness to Work records
- [x] Create display sections for Thermometer Check records
- [x] Update FSAI export to include all new record types
- [x] Test Reports page with all record types

Delivery Records & Voice Recording Fixes

- [x] Fix voice recording bug - stays locked in recording mode and never stops
- [x] Add manual stop recording button with red pulsing indicator
- [x] Add 30-second auto-stop safety timeout
- [x] Add delivery record interface to Reports page
- [x] Load delivery records from localStorage in Reports
- [x] Add delivery summary card to Reports
- [x] Create display section for delivery records with product breakdown
- [x] Add delivery records to FSAI export (SC1 section)

- [x] Add localStorage save functionality to Delivery page
- [x] Test delivery form with voice recording
- [x] Test delivery records display in Reports

Voice Commands for All Forms

- [x] Create reusable voice recording hook (useVoiceRecording)
- [x] Add voice recording button to SC5 Hygiene Inspection form
- [x] Add voice transcription parsing for hygiene checkpoints
- [x] Add voice recording button to SC6 Training

HST Engagement Checklist (Suggested Next Steps)

• Confirm MVP goals for Phase 1 (pilot-ready, not full commercial launch).

• Decide what to keep vs replace from current MVP (especially AI/OCR/voice implementations).

• Define deterministic compliance rules engine boundaries (critical limits, corrective actions, audit trail).

• Confirm data model + tenant model + user roles for pilot.

• Define voice reliability approach: confidence scoring, confirmations, fallbacks, noisy environment handling.

• Define OCR approach: field confidence, manual verification UI, supplier/invoice variation strategy.

• Agree deliverables and milestones (e.g., Phase 0 discovery; Phase 1 hardening; Phase 2 voice/OCR).

• Confirm DevOps plan: hosting, monitoring, backups, logging, incident response for pilot.

Attachments: The full updated codebase is provided separately as a ZIP export (chefvoice_complete_package.zip).