TED UNIVERSITY

Analysis Report
Chef Bono

Team members:

Berkan Gökgöz
Buse Şahin
Yelda Sıla Mumcu
Eren Serdar

1. Introduction

Although modern recipe platforms have extensive access to culinary ingredients, they are insufficient to meet the operational requirements of real-time cooking, such as uniformizing ingredient preparation techniques, tracking the cooking process, and interpreting user actions in context. Although modern recipe platforms have extensive access to culinary ingredients, they are insufficient to meet the operational requirements of real-time cooking, such as uniformizing ingredient preparation techniques, tracking the cooking process, and interpreting user actions in context. To address these limitations, Chef Bono integrates computer vision, natural language processing, voice command recognition, self-timer management, and structured recipe processing within a unified software architecture. The system processes audio-visual streams from cameras and microphones continuously, enabling low-latency analysis of user actions.he system processes audio-visual streams from cameras and microphones continuously, enabling low-latency analysis of user actions. Utilizing a Gemini-based multimsystem processes audio-visual streams from cameras and microphones cously, enabling low-latency analysis of user actions. Utilizing a Gemini-based multimodal AI model, Chef Bono interprets both visual and linguistic inputs to produce decision outputs that are as accurate as possible.

2. Current system

Before Chef Bono, existing applications relied largely on static recipe implementations and user-driven, manual cooking processes. From a software engineering perspective, the current system presents several technical limitations:

2.1 Limited Data Processing Capabilities

Existing systems:

- Lack real-time image processing,
- Unable to detect or interpret user actions, hindering contextual analysis,
- Operational metrics such as cooking levels, color change or ingredient type recognition cannot be verified.

Therefore, users should manage decision-making based solely on personal judgment.

2.2 Insufficient Interaction Models

Traditional recipe applications:

- Only provide text-based content,
- Do not track step progression,
- Offer no voice command handling, camera-based feedback, or sensor-driven interaction.

As a result, the system remains passive and real-time feedback or error prevention mechanisms cannot be provided.

2.3 Lack of Automation

Current solutions require:

- Timers to be initiated manually by the user,
- Ingredient additions or cooking movements to be detected manually, since no automated sensing exists,
- Recipe adjustments to be performed manually, without AI-driven personalization.

This significantly increases the cognitive load and the likelihood of errors, especially in multi-step recipes.

2.4 Insufficient Data Structuring and Storage

Most existing applications:

- Store recipes in unstructured plain-text formats,
- Do not automatically create diet tags (vegan, gluten-free etc.)
- Lack analytical modules to process or adapt recipe content.

Consequently, the current system does not provide essential modern capabilities such as real-time tracking, contextual interpretation, user modeling, AI-assisted decision-making, or automated process control.

3. Proposed system

3.1 Overview

Chef Bono is designed as an intelligent, interactive virtual kitchen assistant that transforms the cooking experience through real-time guidance and computer vision capabilities. The system operates as an active companion rather than a passive recipe reader, utilizing a camera to monitor cooking activities, a microphone to understand voice commands, and text-to-speech technology to provide natural, conversational feedback.

The architecture follows a modular design where Python serves as the central orchestrator, coordinating between several key components: the Gemini API for advanced image and text processing, speech recognition libraries for voice input, Kokoro or alternative TTS services for

voice output, and a local SQLite database for efficient recipe storage. The system retrieves recipes from Cookbooks.com and MediaWiki/Wikibooks, processes them into a structured format, and presents them step-by-step while monitoring the user's progress through computer vision.

What distinguishes Chef Bono from conventional cooking applications is its ability to provide context-aware assistance. Rather than simply displaying instructions, it actively observes the cooking process, sets automatic timers based on detected actions, analyzes cooking temperatures and ingredient preparation quality, and adapts recipes according to user preferences or dietary restrictions in real-time.

3.2 Functional Requirements

FR1: Recipe Retrieval and Management:

The system will retrieve recipes from external sources, convert them into a structured format with new categories, and store them in a local database for efficient querying and step-by-step execution.

FR2: Voice Interaction:

The system will process user voice commands via STT and provide verbal responses via TTS, supporting fully hands-free operation.

FR3: Visual Monitoring and Feedback:

The system will analyze camera input in real time to detect user actions, ingredient status and cooking progress, and provide immediate corrective feedback when necessary or when asked, depending on the selected mode.

FR4: Automatic Timer Management:

The system will automatically start, update and end cooking timers based on detected cooking actions, without requiring manual user input.

FR5: Recipe Customization:

The system will dynamically adapt recipe steps, ingredient quantities and difficulty levels based on user preferences or restrictions.

FR6: Special Dietary Support:

The system will classify recipes using diet labels generated by artificial intelligence and suggest suitable alternatives based on user restrictions.

FR7: Post-Cooking Assistance:

Once the recipe is completed, the system will provide cleaning reminders and post-cooking safety guidance.

3.3 Nonfunctional Requirements

NFR1: Performance:

- The system will process voice commands within 3 to 10 seconds of the completion of the conversation.
- Computer vision analysis (cooking temperature, ingredient size) will be performed within 3 - 10 seconds of the user request.
- The recipe retrieval process will be completed within 3 - 10 seconds under normal network connection conditions.
- The system will support real-time video processing around 1 frame per 3 – 10 seconds

NFR2: Reliability and Availability:

- The system will seamlessly handle API failures and send informative error messages when external services are unavailable.
- The system will recover from crashes without losing timer status or current cooking progress.

NFR3: Usability:

- The voice interface requiring no technical knowledge to operate.
- The system shall provide concise, clear, instructions without overwhelming users.
- TTS output delivered at an appropriate pace and shall be natural-sounding.

NFR4: Scalability:

- The SQLite database will provide retrieval within less than a second, whilst having a few thousand rows to a few million rows. We aim to keep 6 digits to 7 digits number of recipes.
- The system will minimize API calls by caching processed data locally.

NFR5: Privacy and Security:

- Personal user data will not be stored or shared with third parties.
- Users will have clear control over when the camera and microphone are active.
- System logs shall avoid storing raw personal data such as raw audio and image.

NFR6: Legal Compliance:

- The system shall comply with **KVKK (Turkey)** and **GDPR principles** of consent, minimization, transparency.
- The system shall collect and process camera and audio data **only with explicit user consent**.
- Users shall be informed clearly about what data is captured and how it is processed.

NFR7: Maintainability:

- The system will have modular design principles that allow each component to be updated independently.
- The code will be well documented with a clear separation between business logic, API interfaces, and presentation layers.
- The system will use version control and maintain compatibility with specified library versions.

NFR8: Resource Constraints:

- The system shall operate on standard laptop hardware without requiring GPU acceleration
- Memory usage shall not exceed 4GB during normal operation.

3.4 Pseudo requirements

- Chef Bono is a single orchestration component that delegates tasks to internal modules: AI, TTS, STT, Camera, and SQLite.
- The system accepts a user-selected mode (limited surveillance or full-time surveillance) and adapts module behavior accordingly.
- The user can request recipes with dietary/specification filters (e.g., vegan, gluten-free).
- AI module generates or retrieves recipe candidates and constructs SQL queries for the recipe store.
- SQLite module stores and returns recipe metadata and full recipe content on query.
- TTS module converts text responses and recipe steps into audible speech for the user.
- STT module transcribes user speech into text for downstream intent processing.
- Camera module captures periodic images when in full-time surveillance mode and provides images for AI analysis.
- AI module analyzes camera images to detect hazards, missing ingredients, or incorrect steps and emits intervention signals.

- Bono presents recipe selection choices to the user and accepts selection and presentation preferences (all-at-once or step-by-step).
- In step-by-step mode, Bono requests the next step from AI and instructs TTS to speak it; Bono listens for user questions via STT between steps.
- User questions are routed through STT → AI intent/response generation → Bono → TTS for spoken answers.
- Bono supports restart, stop, and continue process controls initiated by the user at any time.
- Bono records timers and step progress when requested and exposes hooks for setting and updating timers.
- All internal module interactions are explicit and traceable so Bono can log actions and decisions for debugging or audit.

3.5 System models

3.5.1 Scenarios

The user starts the application and then follows the following scenarios in the scenario graph:

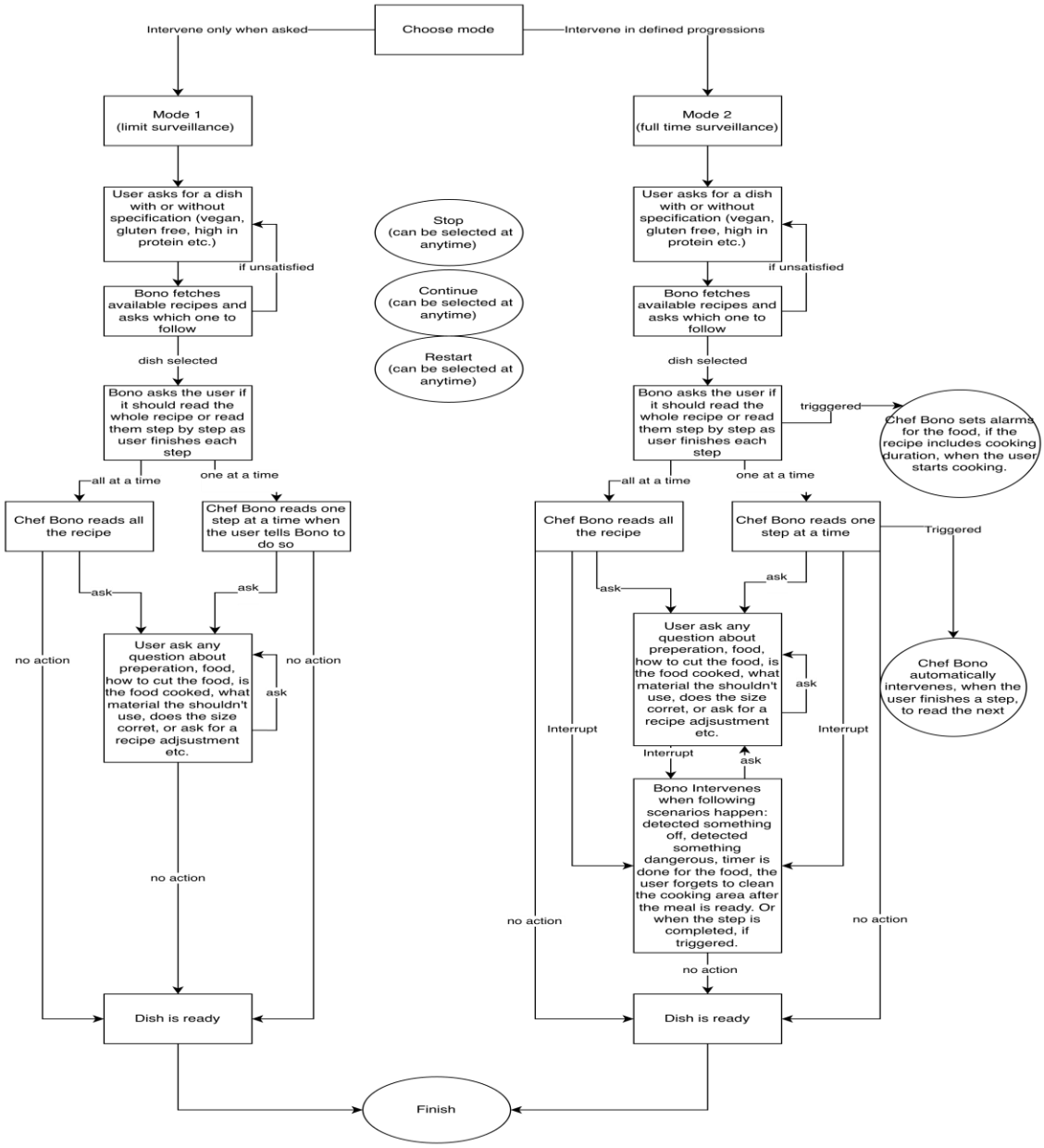
Rectangles are scenarios,

Circles are triggers that add new functionalities to the following scenarios

Ellipses are the application's buttons

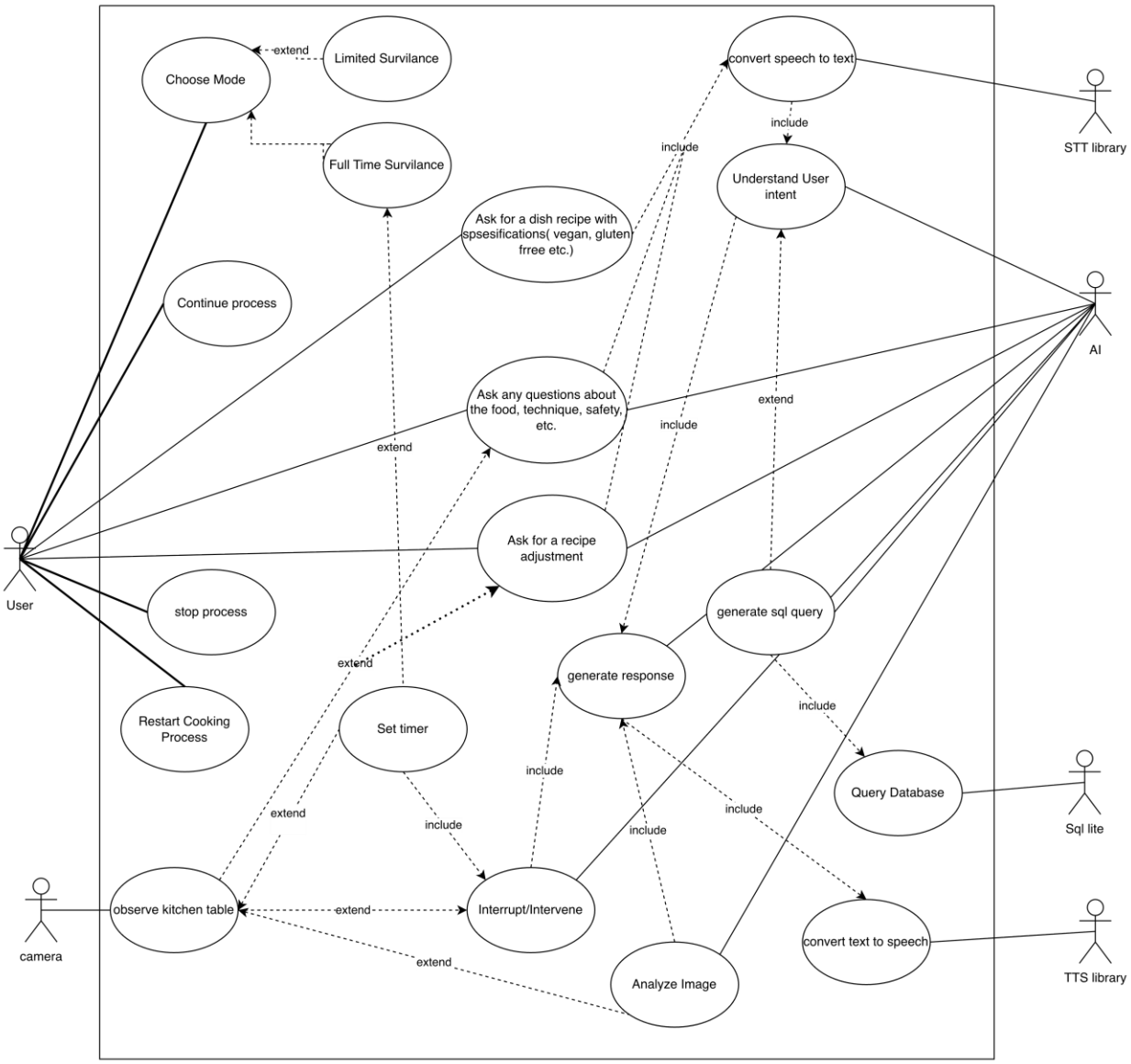
Arrows without a comment are normal scenario flow

Arrows with comments conditional scenario flow



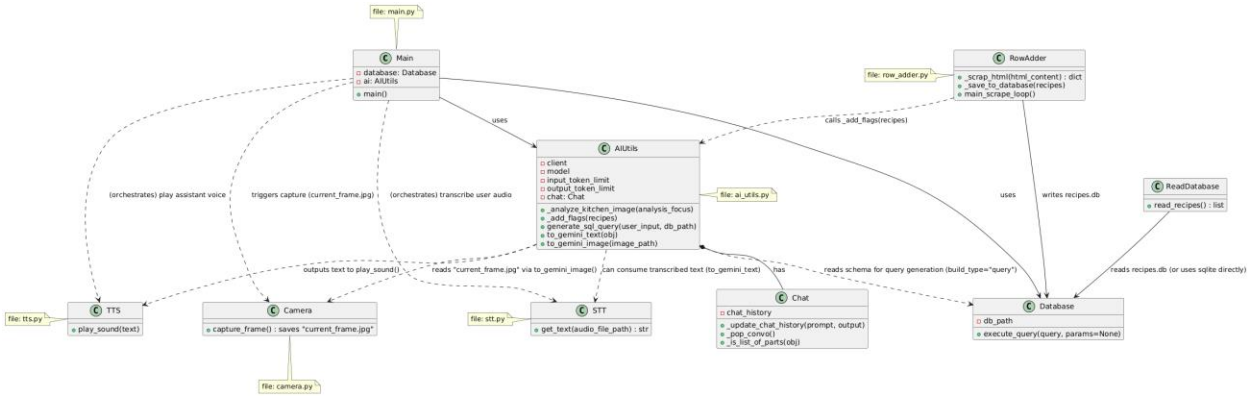
3.5.2 Use case model

Our use case diagram consists of 5 actors. We decided to include external systems as actors given this is a single user app.



3.5.3 Object and class model

Eventhough it's technically not part of the running program, we felt the need to mention another functionality we use to develop the application, which is flag adding. We retrieve open-source recipes and flag them or categorize them, if you will, with AI as vegan, gluten-free, protein score, etc. Then save it to the database.

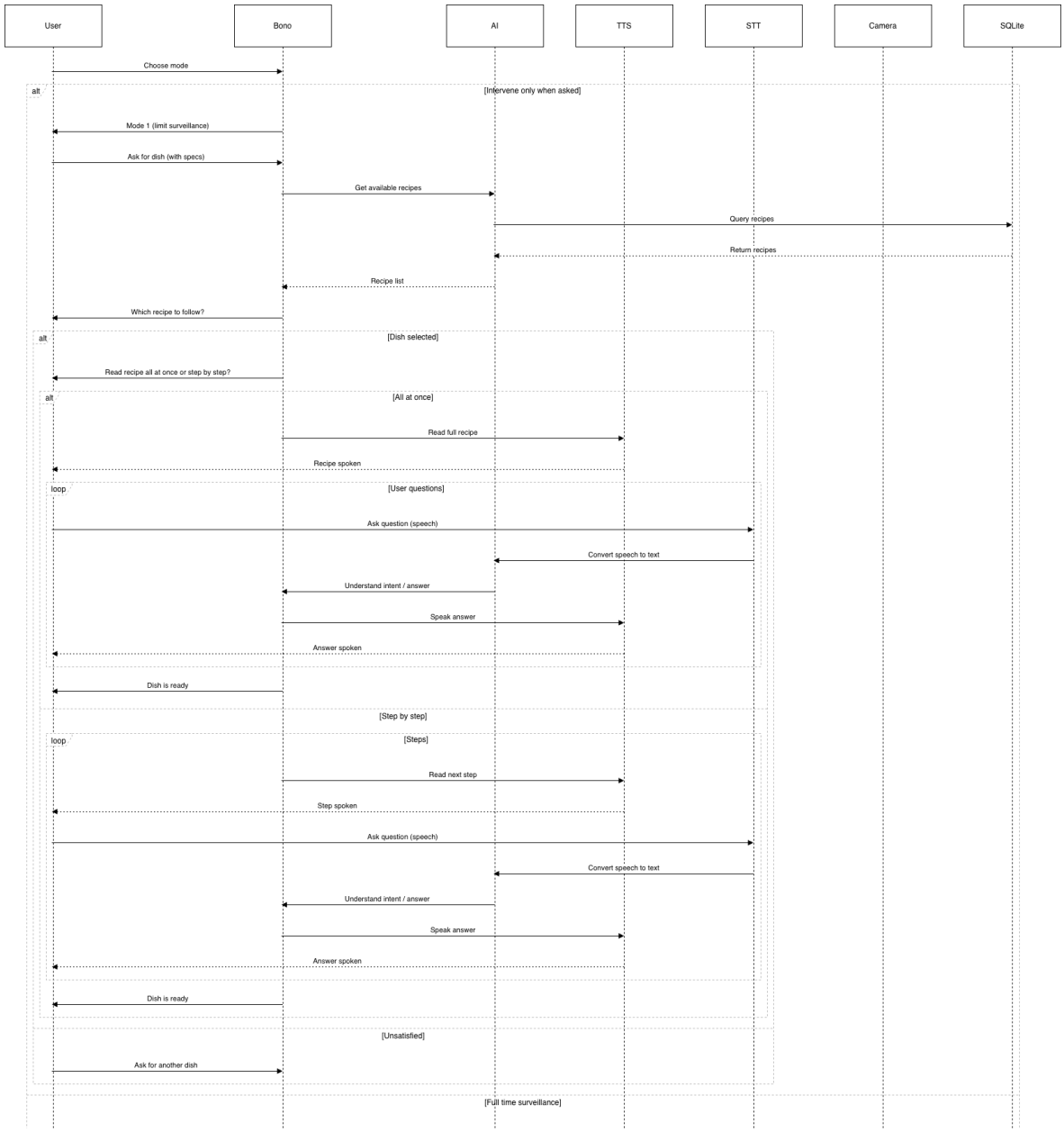


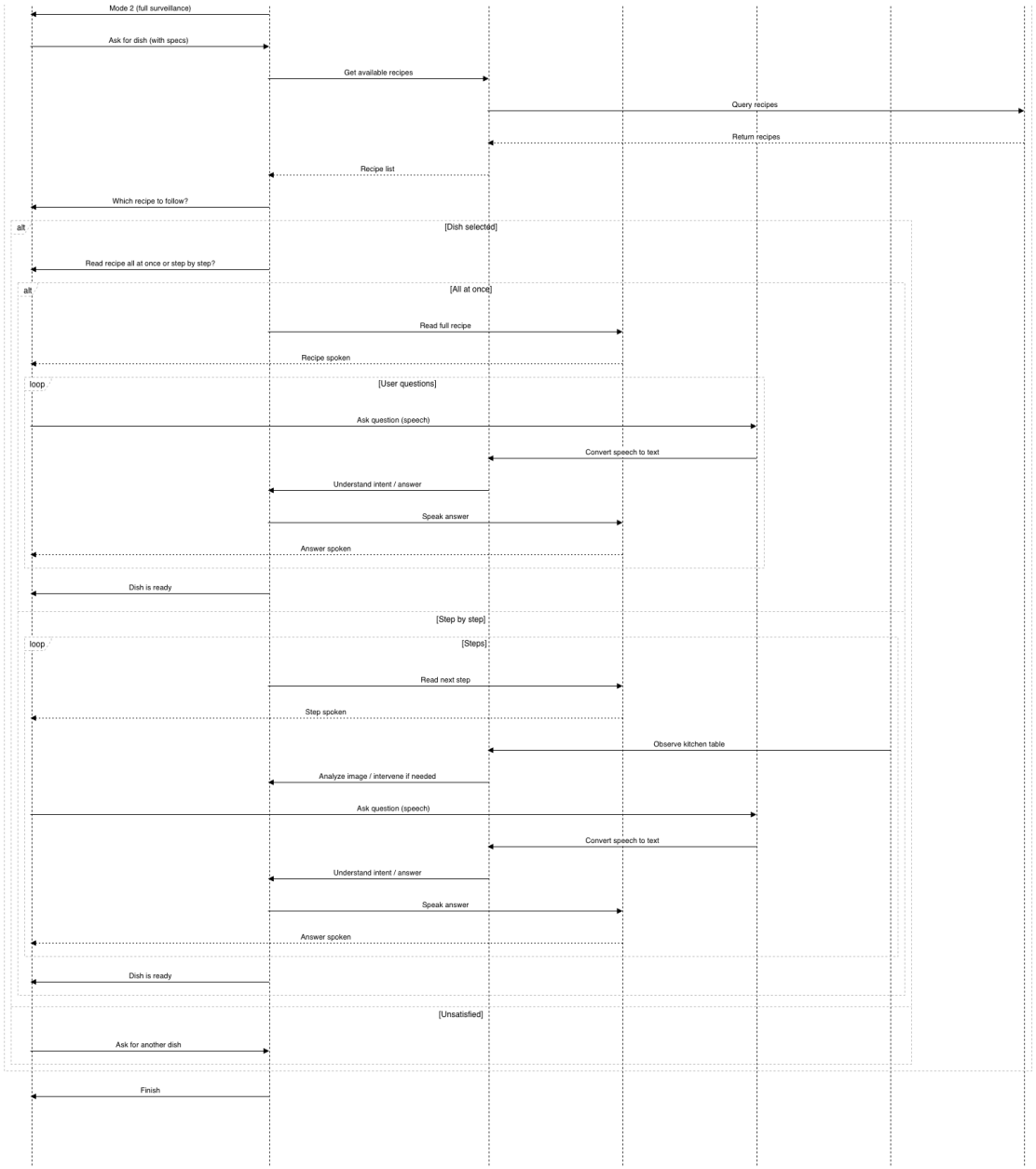
Main: orchestrator — creates Database and AIUtils, triggers image analysis and other high-level flows.
Camera: captures a frame (current_frame.jpg) that AIUtils reads and sends to Gemini for image understanding.
STT (speech-to-text): turns user audio into text; Main or AIUtils sends that text to Gemini.
AIUtils: central AI worker — accepts image + transcribed text, communicates with Gemini, maintains Chat state, generates outputs (text, queries, flags).
Chat: internal to AIUtils — holds conversation history and helps manage context/window limits.
TTS: plays AI-generated text as audio so the user hears the assistant.
Database (db_query_handler.Database): executes SQL and stores/queries recipes; AIUtils can generate SQL or read schema; RowAdder writes new recipes into it.
RowAdder: scrapes web pages for recipes, asks AIUtils to add classification/metrics (flags), then saves validated recipes to the Database.
ReadDatabase: helper to read and pretty-print recipes from the Database.
Data flow (simple):
User speaks -> STT -> text
Camera -> image file
text + image -> AIUtils -> Gemini -> response
AIUtils response -> TTS (play) and/or Database (store/query)
RowAdder scrapes -> AIUtils adds flags -> Database stores

This is the runtime and dependency structure: Camera/STT feed AIUtils; AIUtils uses Chat and Database; TTS and Database consume AIUtils outputs; RowAdder and ReadDatabase interact directly with the Database.

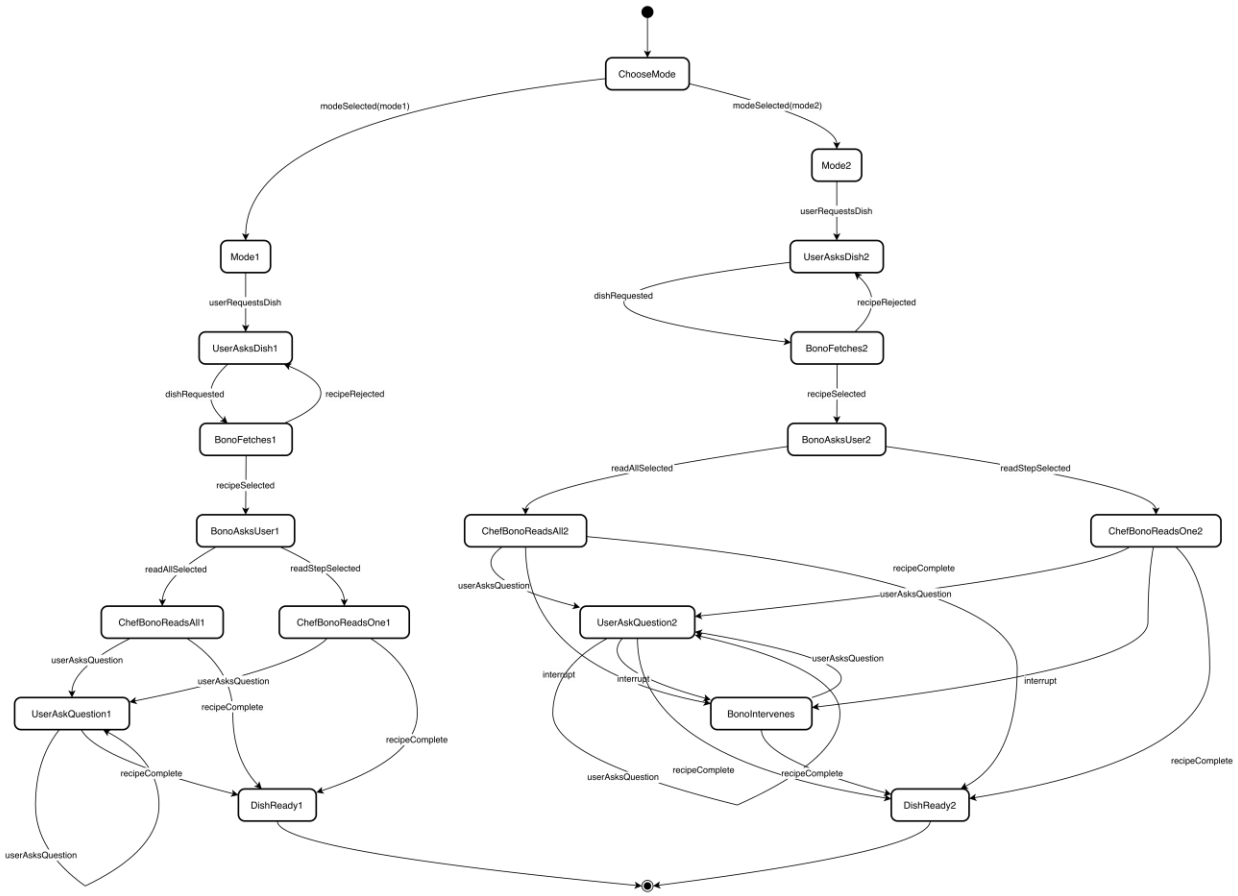
3.5.4 Dynamic models

Sequence diagram

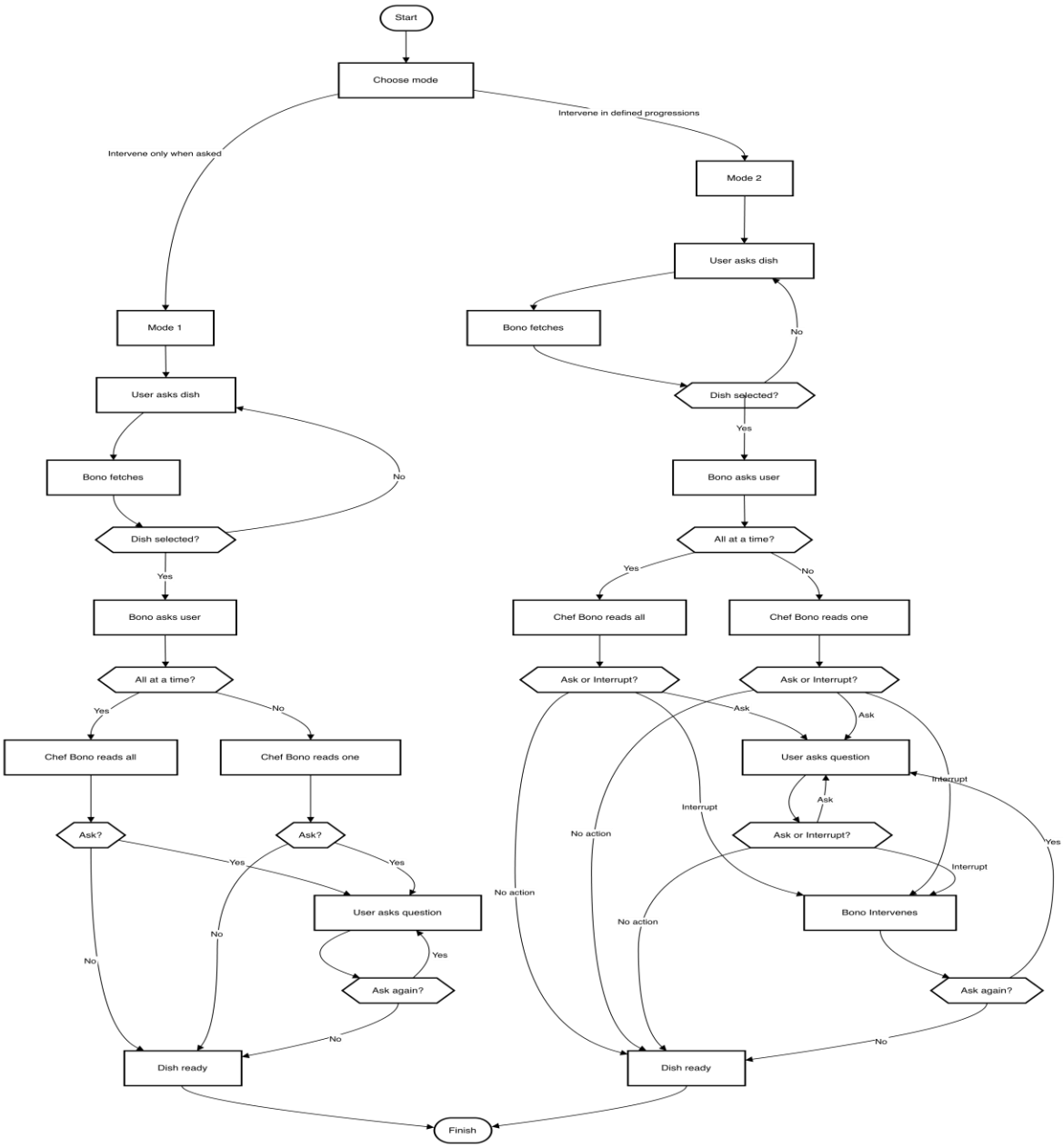

User
Bono
AI
TTS
STT
Camera
SQLite
Choose mode
alt
[Intervene only when asked]
Mode 1 (limit surveillance)
Ask for dish (with specs)
Get available recipes
Query recipes
Return recipes
Recipe list
Which recipe to follow?
alt
[Dish selected]
Read recipe all at once or step by step?
alt
[All at once]
Read full recipe
Recipe spoken
loop
[User questions]
Ask question (speech)
Convert speech to text
Understand intent / answer
Speak answer
Answer spoken
Dish is ready
[Step by step]
loop
[Steps]
Read next step
Step spoken
Ask question (speech)
Convert speech to text
Understand intent / answer
Speak answer
Answer spoken
Dish is ready
[Unsatisfied]
Ask for another dish
[Full time surveillance]



State Diagram:



Activity Diagram:



3.5.5 User interface - navigational paths and screen mock-ups

Given this app is voice communication based, the interface is fairly basic and simple right now.

4. Glossary

- **AI (Artificial Intelligence):**
  A computational system capable of performing tasks that typically require human intelligence, such as image interpretation, language understanding, and decision-making.
- **Multimodal Model:**
  An AI model that processes information from multiple input channels (e.g.,text+image+audio) simultaneously to produce context-aware outputs.
- **Computer Vision (CV):**
  A field of AI that enables systems to interpret and analyze visual information from images or video frames (e.g., detecting cooking progress, cut size, color changes).
- **Speech-to-Text (STT):**
  A component that converts spoken language into written text, enabling voice-command interaction during cooking.
- **Text-to-Speech (TTS):**
  A module that transforms text-based outputs into synthetic speech, allowing the system to provide audible instructions and feedback.

- **Natural Language Processing (NLP):**
  A set of AI techniques used to interpret and generate human language, enabling understanding of user intents and conversational interaction.
- **SQLite Database:**
  A lightweight, file-based relational database used for storing structured recipes, metadata, and AI-generated tags locally.
- **Recipe Flags / Metadata:**
  AI-generated attributes, such as vegan, gluten-free, high-protein, or difficulty level, assigned to recipes for classification and filtering.
- **Modular Architecture:**
  A software design approach in which system components are separated into independent modules (e.g., STT engine, vision processing, database manager) to improve maintainability and scalability.
- **Real-Time Processing:**
  The ability to analyze audiovisual input with minimal latency, enabling immediate feedback during the cooking workflow.
- **TimerAutomation:**
  A mechanism that detects user actions (e.g., ingredients added to a pan) and automatically starts or adjusts timers without manual input.

5. References

- **TheMealDB API Documentation.**
  Available at: https://www.themealdb.com/api.php
- **Cookbooks.com – Recipe Database.**
  Available at: https://www.cookbooks.com/
- **MediaWiki / Wikibooks – Cookbook Project.**
  Available at: https://en.wikibooks.org/wiki/Cookbook
- **Google Gemini API Documentation.**
  Available at: https://ai.google.dev/gemini-api
- **OpenCV – Computer Vision Library Documentation.**
  Available at: https://docs.opencv.org/
- **Python SpeechRecognition Library Documentation.**
  Available at: https://pypi.org/project/SpeechRecognition/
- **Kokoro / Coqui TTS Documentation.**
  Available at: https://coqui.ai/
- **SQLite Official Documentation.**
  Available at: https://www.sqlite.org/docs.html
- **Python 3 Standard Library Documentation.**
  Available at: https://docs.python.org/3/library/
- https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-introduction