# **טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)**

**הגשת פרויקט**

**<Carebells> < B22><18>**

| **שם חבר.ת הצוות** | **תז** |
| --- | --- |
| לאון פלדמן | 207434879 |
| מור הודיה ממן | 206692592 |
| אשרף עטשי | 211622162 |
| חסן עבדאללה | 325814432 |
| כיאן גדבאן | 211613708 |
| עומרי עבדאללה | 323964817 |

**קישור לגיט:** <https://github.com/Leontarin/CareBell>

**קישור לאתר:** <https://carebell.vercel.app>

1. **מהנדס המערכת: לאון פלדמן**העבודה חולקה לזוגות (כפי שניתן לראות בטבלה), כל זוג עבד על סעיפים מסויימים מתוך המשימה, ולבסוף עברנו כולנו יחד על הכל, אישרנו והגשנו. התנהלנו בקבוצת הוואטסאפ, בעזרת גוגל drive וכל העבודה על הפרוייקט נעשתה תחת גיטהאב משותף.

| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| --- | --- | --- |
| לאון פלדמן  מור הודיה ממן | 6,7 | הושלם |
|
| חסן עבדאללה  עומרי עבדאללה | 2,3 | הושלם |
|
| אשרף עטשי  כיאן גדבאן | 4,5 | הושלם |
|

**2. רשימת דרישות:**

* **דרישות פונקציונליות**
* **דרישות לא פונקציונליות (בנפרד, יש לסווג דרישות לא פונקציונליות לפי wikipedia NFR).**
* **דרישות ממשק חיצוניות.**

# **Software Requirement Specification (SRS) - CareBell**

## **Table of Contents**

1. Introduction
2. System and Functional Requirements
3. External Interface Requirements
4. Non-Functional Requirements

## **1. Introduction**

### **1.1 Product Scope**

CareBell is a digital care assistance platform designed specifically for seniors participating in the "Meals on Wheels" program. The system enhances quality of life for elderly users by providing personalized digital assistance, medication reminders, social connectivity, and meal information management through an intuitive, senior-friendly interface.

**Business Goals:**

* Reduce social isolation among seniors by 40%
* Improve medication adherence rates by 60%
* Increase program participant satisfaction by 35%
* Bridge the digital divide for elderly users

**Benefits:**

* Enhanced health management through automated reminders
* Improved social connectivity with family and community
* Simplified access to meal and nutritional information
* Emergency support accessibility

### **1.2 Product Value**

CareBell addresses critical challenges faced by seniors:

**Problem Solved:** Social isolation and health management difficulties among elderly participants in community meal programs

**Value Proposition:**

* **Health Management**: Automated, personalized medication tracking and reminders
* **Social Connection**: Simplified video calling and virtual community participation
* **Information Access**: Easy-to-understand meal information and daily updates
* **Emergency Support**: One-touch access to emergency contacts and services
* **Digital Inclusion**: Technology designed specifically for limited tech experience

### **1.3 Intended Audience**

**Primary Users:**

* Adults aged 70+ participating in "Meals on Wheels" programs
* Limited technology experience (beginner level)
* Potential visual or hearing impairments
* Living independently but requiring daily assistance
* Varying cognitive function levels

**User Characteristics:**

* Prefer large, simple interfaces
* Need audio support for visual content
* Require minimal learning curve
* Value routine and consistency

**Secondary Users:**

* Family members and caregivers
* Program administrators
* Healthcare providers

### **1.4 Definitions and Acronyms**

**SRS** - Software Requirement Specification

**NFR** - Non-Functional Requirement

**API** - Application Programming Interface

**QR Code** - Quick Response Code for scanning meal package information

**WebRTC** - Web Real-Time Communication technology for video calls

**UI/UX** - User Interface/User Experience

**Bella** - AI assistant avatar within the CareBell system

**HIPAA** - Health Insurance Portability and Accountability Act

**AES-256** - Advanced Encryption Standard with 256-bit key

## **2. System and Functional Requirements**

### **2.1 User Interface and Navigation**

**FR001:** The system must display large, high-contrast buttons with minimum 60px height for all primary actions

**FR002:** The system must provide a clearly labeled "Back to Main Menu" button on every screen except the homepage

**FR003:** The system must include simple breadcrumb navigation showing current location within the app

**FR004:** The system must support voice-activated navigation using predefined voice commands

**FR005:** The system must provide text-to-speech capability for all on-screen content

**FR006:** The system must include adjustable text size settings with four levels: Small, Medium, Large, Extra Large

**FR007:** The system must display no more than 6 primary actions on the main screen to prevent cognitive overload

**FR008:** The system must use consistent color schemes and visual language throughout the application

**FR009:** The system must provide immediate visual feedback for all user interactions

**FR010:** The system must include a persistent red "Emergency" button accessible from any screen

### **2.2 Bella AI Assistant Functionality**

**FR011:** The system must include an AI avatar named "Bella" with context-appropriate animated facial expressions

**FR012:** The system must allow users to initiate conversations with Bella via large button click or voice command

**FR013:** The system must enable Bella to provide personalized medication reminders with user's name

**FR014:** The system must allow Bella to answer questions about meal contents and nutritional information

**FR015:** The system must enable Bella to provide weather updates and simplified news summaries

**FR016:** The system must allow Bella to guide users through app features with step-by-step audio instructions

**FR017:** The system must enable Bella to recognize basic emotional cues and respond appropriately

**FR018:** The system must allow Bella to maintain conversation context within a single user session

**FR019:** The system must enable Bella to escalate to human support when unable to assist effectively

**FR020:** The system must log all Bella interactions for continuous improvement of response accuracy

### **2.3 Medication Management**

**FR021:** The system must allow users to input medication schedules with dosage and timing information

**FR022:** The system must send both visual and audio reminders at prescribed medication times

**FR023:** The system must enable users to mark medications as "taken," "skipped," or "delayed"

**FR024:** The system must track medication adherence patterns and generate weekly reports

**FR025:** The system must alert users when medication supplies are projected to run low (7-day warning)

**FR026:** The system must provide basic drug interaction warnings when multiple medications are entered

**FR027:** The system must allow authorized family members to receive medication adherence notifications

**FR028:** The system must include a pill identification feature using image recognition technology

**FR029:** The system must provide medication education materials written in simple, large-font language

**FR030:** The system must integrate with participating pharmacy systems for automatic refill reminders

### **2.4 Social Connection Features**

**FR031:** The system must support one-click video calling to pre-configured family contacts

**FR032:** The system must enable group video calls with up to 6 participants simultaneously

**FR033:** The system must provide virtual meeting rooms for scheduled social activities

**FR034:** The system must allow users to join community events with simple calendar integration

**FR035:** The system must include a messaging system supporting large text and voice messages

**FR036:** The system must enable photo sharing with automatic image resizing and simple navigation

**FR037:** The system must provide calendar integration showing upcoming family events and medical appointments

**FR038:** The system must include cognitive stimulation games designed for seniors

**FR039:** The system must enable users to create and manage their personal contact list

**FR040:** The system must provide activity suggestions based on user preferences and health status

### **2.5 Meal Information System**

**FR041:** The system must scan QR codes on meal packages to display comprehensive nutritional information

**FR042:** The system must highlight allergens and dietary restrictions relevant to the user's health profile

**FR043:** The system must provide meal preparation instructions in large, clear text with visual aids

**FR044:** The system must display caloric and nutritional content in easy-to-understand visual formats

**FR045:** The system must allow users to rate meals and provide simple feedback

**FR046:** The system must track user's meal preferences and dietary restrictions over time

**FR047:** The system must provide alternative meal suggestions based on dietary needs and preferences

**FR048:** The system must include educational content about nutrition specifically for seniors

**FR049:** The system must integrate with meal delivery schedules and provide arrival notifications

**FR050:** The system must store meal history for nutritional tracking and health reporting

## **3. External Interface Requirements**

### **3.1 User Interface Requirements**

**EIR001:** The system must provide a touch-optimized interface with minimum 44px touch targets

**EIR002:** The system must support both portrait and landscape orientations on tablet devices

**EIR003:** The system must be compatible with screen readers and assistive technologies

**EIR004:** The system must provide high contrast mode with 4.5:1 minimum contrast ratio

**EIR005:** The system must support keyboard navigation for all interactive elements

### **3.2 Hardware Interface Requirements**

**EIR006:** The system must interface with tablet cameras for QR code scanning functionality

**EIR007:** The system must utilize device microphones for voice input with noise cancellation

**EIR008:** The system must access device speakers for audio output with volume normalization

**EIR009:** The system must support standard Bluetooth connectivity for hearing aid integration

**EIR010:** The system must function on tablets with minimum 3GB RAM and Android 8.0+ or iOS 12+

### **3.3 Software Interface Requirements**

**EIR011:** The system must integrate with external weather APIs (OpenWeatherMap) for current conditions

**EIR012:** The system must connect to news RSS feeds filtered for senior-relevant content

**EIR013:** The system must implement WebRTC for peer-to-peer video communications

**EIR014:** The system must integrate with SMS gateways for emergency notifications

**EIR015:** The system must interface with pharmacy APIs for medication verification and refill management

**EIR016:** The system must connect to calendar services (Google Calendar, Outlook) for appointment management

### **3.4 Communication Interface Requirements**

**EIR017:** The system must use HTTPS/TLS 1.3 for all data transmission

**EIR018:** The system must implement WebSocket connections for real-time chat and notifications

**EIR019:** The system must support RESTful API architecture for data exchange

**EIR020:** The system must integrate with SMTP servers for email notifications

### **3.5 Database Interface Requirements**

**EIR021:** The system must interface with MongoDB for user profile and preference storage

**EIR022:** The system must support database connection pooling for optimal performance

**EIR023:** The system must implement automated database backup and recovery mechanisms

**EIR024:** The system must ensure ACID compliance for critical health-related transactions

**EIR025:** The system must provide data export capabilities for user information portability

## 

## **4. Non-Functional Requirements**

### **4.1 Performance Requirements**

**NFR001:** The application must load the main screen within 3 seconds on all supported devices

**NFR002:** The system must support up to 1000 concurrent users without performance degradation

**NFR003:** Voice command response time must not exceed 2 seconds

**NFR004:** Video calls must maintain quality with maximum 200ms latency

**NFR005:** Database queries must return results within 1 second for 95% of requests

### **4.2 Reliability Requirements**

**NFR006:** The system must provide 99% uptime for core services such as reminders and communication features

**NFR007:** The system must perform automatic backup of user data every 24 hours

**NFR008:** The system must recover from failures within 15 minutes of issue detection

**NFR009:** The system must provide graceful degradation when external services are unavailable

**NFR010:** The system must maintain service during planned maintenance with less than 1 hour downtime per month

### **4.3 Usability Requirements**

**NFR011:** The system must offer a senior-friendly user interface with large buttons, clear labels, and readable text

**NFR012:** The application must fully support screen readers and voice commands

**NFR013:** The system must include a "night mode" with high contrast for improved visibility

**NFR014:** The system must allow font size adjustment at four different levels

**NFR015:** New users must be able to complete basic tasks within 10 minutes of first use

### **4.4 Security Requirements**

**NFR016:** The system must encrypt all sensitive user data using AES-256 encryption

**NFR017:** The system must authenticate user identity before accessing medical information

**NFR018:** The system must log all user actions for security audit purposes

**NFR019:** The system must comply with HIPAA regulations for health information protection

**NFR020:** The system must implement multi-factor authentication for administrative access

### **4.5 Maintainability Requirements**

**NFR021:** The system must use modular code architecture to allow easy updates and feature additions

**NFR022:** The system must include comprehensive logging for issue identification and resolution

**NFR023:** The system must support automated testing with minimum 80% code coverage

**NFR024:** The system must provide clear documentation for all APIs and system components

**NFR025:** Code deployment must be automated with rollback capabilities within 5 minutes

### **4.6 Compatibility Requirements**

**NFR026:** The system must function optimally on tablets and desktop computers

**NFR027:** The system must be compatible with major web browsers (Chrome, Firefox, Safari, Edge)

**NFR028:** The system must support responsive design for different screen sizes (10-15 inch displays)

**NFR029:** The system must work with assistive technologies including screen magnifiers

**NFR030:** The system must support offline functionality for core features during internet outages

### **4.7 Scalability Requirements**

**NFR031:** The system architecture must support horizontal scaling to accommodate growing user base

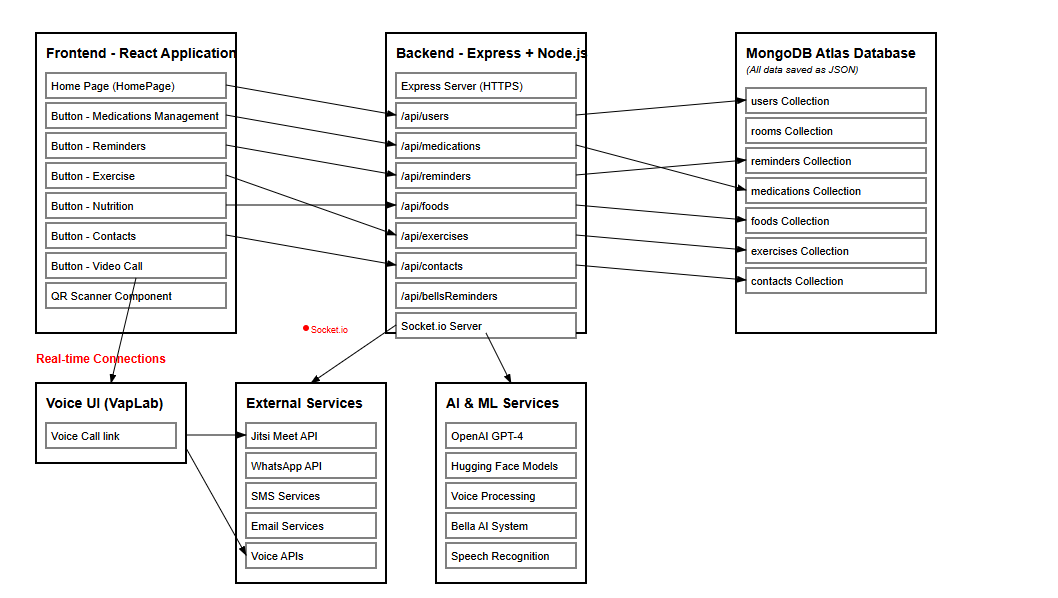
**NFR032:** The database must handle up to 10,000 user records efficiently

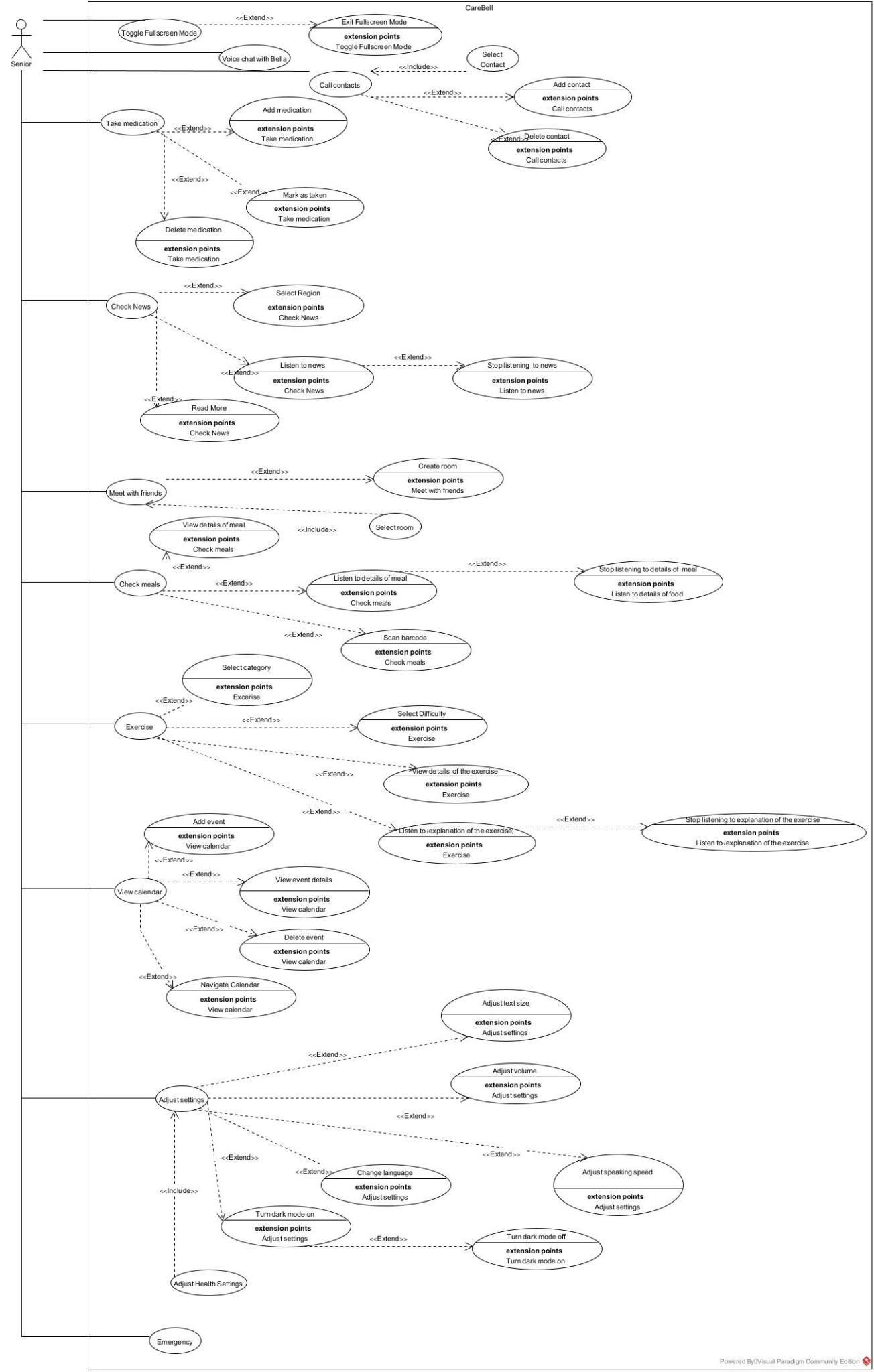
**NFR033:** The system must support geographic distribution across multiple server locations

**NFR034:** The system must handle peak loads of 5x normal usage during emergency situations

**NFR035:** The system must scale video calling infrastructure to support 500 concurrent calls

**3. הציגו ארכיטקטורה מעודכנת של האתר (תרשים הכולל את האלמנטים המרכזיים).**



**4. הציגו דיאגרמת use case המתארת את השימוש באתר.**.

5**. יש להציג מבנה סופי של האתר שלכם:**

**נדרשת בכל פרויקט פריסה מלאה (deploment) של הפרויקט! ב - vercel.  
לא יתקבלו הגשות של קבצים או אתר ב -localhost .**

**טכנולוגיות:    להלן המרכיבים הטכנולוגיים המומלצים לשימוש בפרויקט:**

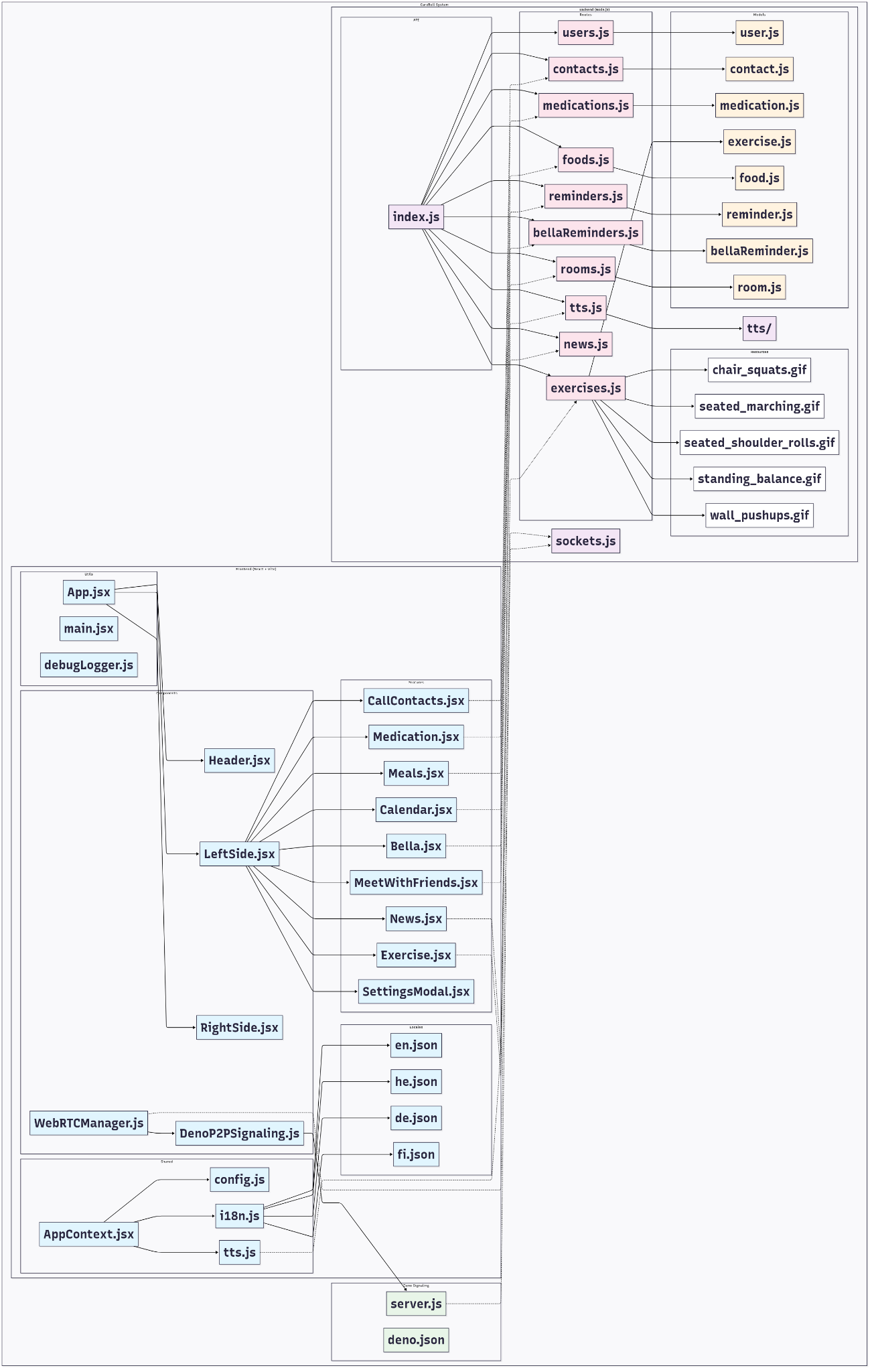
**·           - front-end: React/Preact/Next with Tailwind**

**- back-end:**

**- option 1 - remote services/APIs**

**- option 2 - node js/express deployed on remote web server**

**א. האתר ימומש ב -React/Preact/Next, וכן שימוש ב Tailwind - נא להציג דיאגרמה המתארת את התיקיות והקבצים השונים. יש לפרט את הקומפוננטות השונות.**



**ב.** יש לפרט את פריטי המידע - יש להשתמש במידע אמיתי ורלוונטי לפרויקט שלכם (בשליפה ממסד נתונים חיצוני או מ - API). יש להראות דיאגרמת מבנה DB.

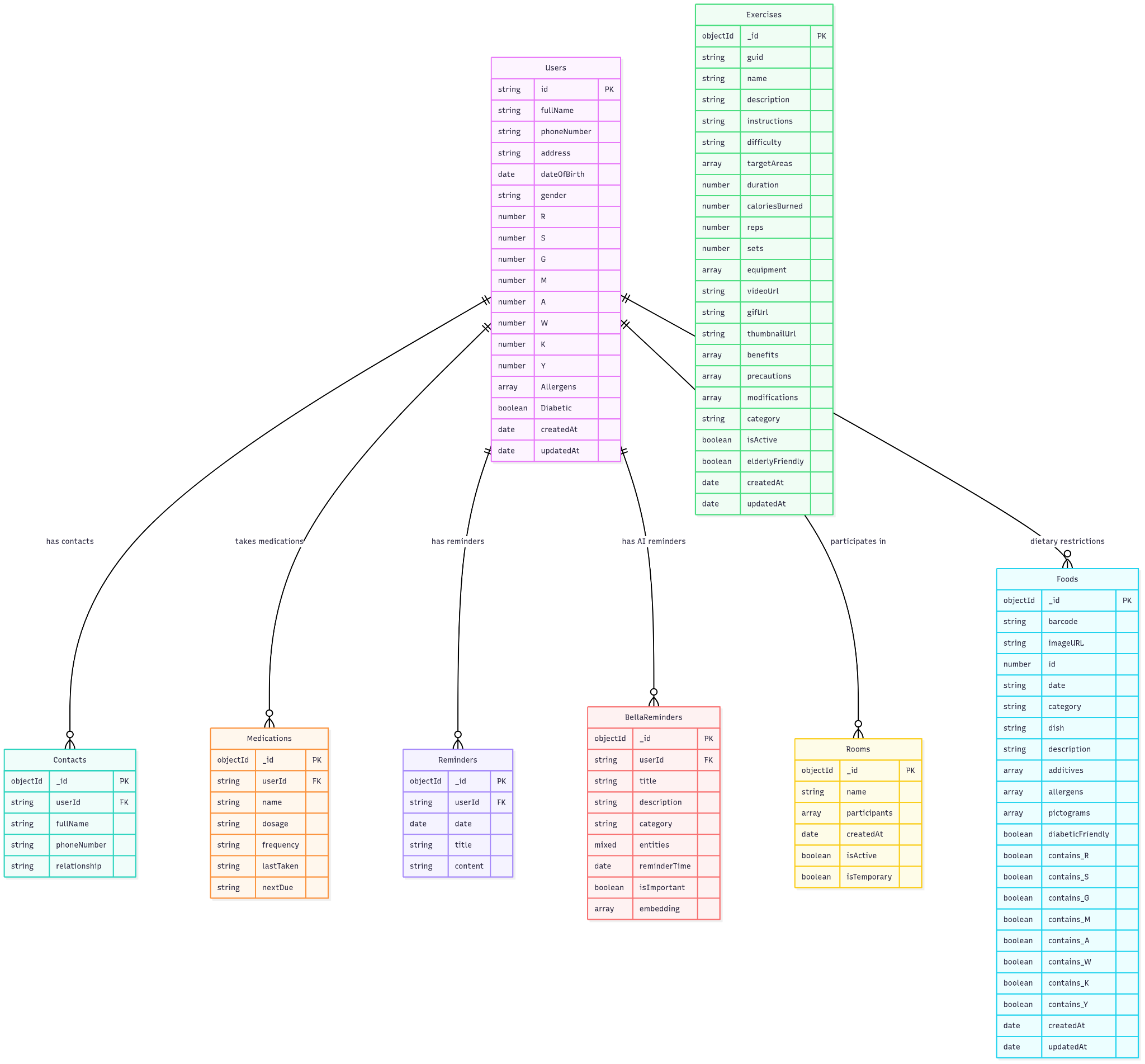
* משתמשים
  + מידע אישי: שם, טלפון, כתובת, תאריך לידה, מגדר
  + אלרגיות ומגבלות תזונה (R, S, G, M, A, W, K, Y)
  + סוכרת (Diabetic)
* אנשי קשר
  + שם מלא, טלפון, קרבה משפחתית
* תרופיות
  + שם התרופה, מינון, תדירות, לקיחה אחרונה, מועד הבא
* מזון
  + ברקוד, תמונה, קטגוריה, מנה, תיאור
  + תוספי מזון, אלרגנים, מתאים לסוכרתיים
* אימונים
  + שם, תיאור, הוראות, רמת קושי
  + אזורי מטרה, משך, קלוריות, חזרות
  + קישורי וידאו/GIF, מתאים לקשישים
* תזכורות
  + תאריך, כותרת, תוכן
* תזכורות בלה
  + קטגוריה, ישויות מחולצות, חשיבות
  + עיבוד AI של מידע אישי
* חדרים
  + שם, משתתפים, סטטוס פעיל, זמני/קבוע

השתמשנו ב MongoDB Atlas כמסד ניתונים ענני

השתמשנו ב Tagesschau API כדי הציג חדשות גרמניות אמיתיות

השתמשנו ב OpenAI API לעיבוד טקסט לתזכורות חכמות

השתמשנו ב [Socket.IO](http://socket.io) לתקשורת real-time



קשרים במסד ניתונים:

* Users → Contacts: קשר 1:N (משתמש יכול לקבל מספר אנשי קשר)
* Users → Medications: קשר 1:N (משתמש יכול לקבל מספר תרופות)
* Users → Reminders: קשר 1:N (משתמש יכול לקבל מספר תזכורות)
* Users → BellaReminders: קשר 1:N (משתמש יכול לקבל תזכורות AI)
* Users → Rooms: קשר N:N (משתמש יכול להיות במספר חדרים)
* Users → Foods: קשר לוגי (מגבלות תזונה)

**Developer Documentation for Carebell (תיק למתכנת)**

# **Project Overview**

CareBell combines a **React (Vite) frontend** with an **Express+MongoDB backend** and an **Deno WebSocket signaling server** for peer‑to‑peer video calling used in Meet-With-Friends.  
All applications are run using nodejs  
  
Inside the git repository all environments (front/back/deno) are divided into folders.  
Running them in Vercel or Deno requires setting that folder as root.

**CareBell**/ - The CareBell frontend React application.  
**backend/** - The expressJs server that contains REST API and MongoDB api

**deno-signaling/** - Deployment for Meet-With-Friends P2P signaling using Deno

# **Deno Signaling Server Overview**

The P2P signaling service in deno-signaling/server.js designed to run on Deno deployment creates a WebSocket for WebRTC, it is responsible for maintaining active rooms and routes ICE candidates/offers/answers between the peers. The server exposes /health and /stats HTTP endpoints for status retreival

# **Backend Overview**

The backend runs on expressJs and is the main way the React app gets data.  
It currently supports Vercel deployment and generic nodejs (npm) deployments  
We’ll go over each folder inside backend/ and what does it include

**backend/api/** - contains the index.js that starts the server, both the vercel.json and package.json look at the index.js in this folder  
 **backend/models/**  - contains all the models used for mongodb, when adding or retrieving objects from mongodb, it will use the models as templates.  
  
**backend/routes/** - expressJs REST API, we expose these to provide different services using the server.  
They can be called from the frontend or from your browser by typing the server URL and the exposed REST API  
(e.g. https://URL.COM/users/addUser alongside a json with a fitting json object with the user model)  
Below is a table of each route and what service does it provide

| Route Name | Filename | Description |
| --- | --- | --- |
| Users | users.js | User profile CRUD |
| Contacts | contacts.js | Manage phone contacts (Add/Remove) |
| Foods | foods.js | Fetch meals from the DB |
| Medications | medications.js | Manage medications per user CRUD |
| Reminders | reminders.js | Calendar reminders per user CRUD |
| Exercises | exercises.js | Fetch exercises from DB |
| Rooms | rooms.js | Create/join/leave video rooms |
| News | news.js | Fetch daily news via Tagesschau API |
| BellaReminders | bellaReminders.js | Analyze text from the Bella AI using OpenAI, extract personal information and store in DB |
| TTS | tts.js | Local TTS api to generate speech audio files from given text in json |

**backend/resources/** - used to save resources locally, like images, currently used to load Exercises gifs.

**backend/tts**/ - contains the binary program for piper, which is what we use for our tts

# **Frontend Overview**

The main application run in React using Vite.  
Root directory is CareBell/  
All the main files and components are under CareBell/src  
The Process when running the app is  
Nodejs -> /index.html -> src/index.jsx -> src/App.jsx  
Then all the other components are loaded underneath App.jsx  
The app is divided into the following folders in CareBell/src:  
**components/ -** has the main Layout files “Header.jsx, LeftSide.jsx, RightSide.jsx” alongside “DenoP2PSignaling.js” and “WebRTCManager.js” used for the video chat room connections  
  
**features/** - Contains all the main features of the application, all features self-contain their DOM and Logic.  
Below is a table explaining what are the existing features and their descriptions

| Feature | Description |
| --- | --- |
| Bella.jsx | Integrates Vapi AI Assistant, the user can chat with the AI and interact with other components, handles user intent, voice call control and chat history (every language uses a different assistant) |
| CallContacts.jsx | Call Contacts interface and management, each button is set to call the defined telephone using a tel url |
| MeetWithFriends.jsx | Video rooms, create/join/leave video chat rooms with WebRTC, uses DenoP2PSignaling and WebRTCManager |
| Medication.jsx | Medication manager, list medications, mark as taken |
| Meals.jsx | Scan QR Codes or type manually to fetch the meal data from the DB, utilizes the TTS in the backend. |
| News.jsx | Fetch daily news from the backend, utilizes TTS from the backend |
| Exercise.jsx | Fetch exercises from backend and show them, utilizies TTS from the backend |
| Calendar.jsx | Simple calendar feature, manages events and reminders. |
| SettingsModal.jsx | Settings screen, adjust font size, language. And set health options (e.g allergens) Unimplemented features: AI speech volume and speed Currently users are picked manually from a list here. |

**locales/** - most of the features use locales to display the text in the set language in SettingsModal, therefore we utilize i18n (set as ‘t’ in code) to display the text from the correct locale files, currently there are 4 supported languages:  
English - en.json  
German - de.json  
Hebrew - he.json  
Finnish - fi.json

**resources/**  - contains resources for the frontend, currently houses the Bella AI image and the CareBells logo  
  
**shared**/ - between some features/components we want to share values or make some values public to the app itself, we keep everything related to that here.  
AppContext.jsx - contains React Context that can be used between features (like user, almost every feature needs to know what user it is on right now)  
config.js - config file for backend API url or other features like P2P, News.  
i18n.js- we implement i18next from this file, the locales are set here and all the components in the app use this file to display language-locale-based text  
tts.js **-** simple script that features can call to get an immediate TTS response from the backend  
  
**utils**/ - utilities and useful tools for the developer, currently houses debugLogger.js, which was mainly used to test new features and log them into the console so we can debug.

# **API and External Services**

Throughout the backend and frontend we utilize API calls and some external services, below will be explained what each service is, where is it used, what .env /Environment Variables/API keys we need to run it.  
  
MongoDB via Mongoose - Database connected from our expressJs backend,  
Connection URL: [mongodb+srv://CareBell:vTDHDu9pHns9HNlw@cluster0.bqe7zge.mongodb.net/CareBell](mailto:vTDHDu9pHns9HNlw@cluster0.bqe7zge.mongodb.net)  
Under the connection we use the database ‘CareBell’ where all the collections are saved.  
We require the mongodb url as an environment variable:  
MONGODB\_URI  
  
VAPI AI - <https://vapi.ai/>  
We use VAPI as an API call to start a call with our AI assistant, inside the site we set assistants, their AI language model, their TTS service and transcriber so we can display the text to the user.  
Since we use multiple languages we set it so for each language we have a different assistant alongside the account’s public key for vapi  
Environment Variables (Set in both .env and vercel)  
VITE\_VAPI\_PUBLIC\_KEY

VITE\_VAPI\_ASSISTANT\_ID\_EN

VITE\_VAPI\_ASSISTANT\_ID\_DE

VITE\_VAPI\_ASSISTANT\_ID\_HE

OpenAI - <https://platform.openai.com>  
in our backend we can analyze the text provided by VAPI AI to analyze whether the user said personal information that should be remembered so we can put in the database later, for this we utilize OpenAI API calls from the backend,  
Currently we use OpenAI GPT-3.5 with a prompt to ask whether a text is personal information, for that we require an OpenAI Environment Variable:  
OPENAI\_KEY  
  
Tagesschau API (for News) - <https://www.tagesschau.de/api2u/news>  
 To fetch our daily news we utilize a free API service called Tagesschau.  
It’s simple to implement and doesn’t require anything  
ReadMe/Usage url: <https://github.com/AndreasFischer1985/tagesschau-api/blob/main/README_en.md>

TTS (Piper)- <https://github.com/rhasspy/piper>  
For our manual TTS that does not use VAPI AI, we utilize piper.  
A binary program that uses voice models alongside text to provide an audio output of the text. We save this program manually and depending on whether we are running the program on Vercel/Linux/Windows it automatically picks the correct binary to produce the audio file output

Deno Signaling Server  
a WebSocket/P2P Signaling server that we use for Meet-With-Friends  
It runs on Deno’s standard library using serve from their [server.ts](https://deno.land/std@0.208.0/http/server.ts) with all the tasks defined in deno.json

OpenWeatherMap API **-** <https://api.openweathermap.org/data/2.5/weather>  
To provide the weather/location in the header we use an api by openweathermap,   
It does require a PUBLIC API KEY.

# **AI Prompts and External Code References** During our development we used AI tools to quickly get started on the app, Things like the initial react standard setup “Can you give me code to start a react project, I would like the app to have a components folder where I can later add more features” “Can you give me a proper layout for left side and right side of the screen” And many more that help us get started on the React app itself and other components without spending a lot of time on the technicalities of how to write them specifically, we modified the results to suit our needs better and changes were made. We did use some external code to make VAPI work, most of the code that uses API’s of other services were provided by the documentation, for example the entire process behind sending VAPI information about the user was provided using vapi.send() which is documented here: <https://docs.vapi.ai/assistants/background-messages> Below is a table of all external packages we used and links to their respective docs of usage

| **Package** | Where was it used | Documentation |
| --- | --- | --- |
| **mongoose** | Backend MongoDB models | [mongoosejs.com/docs](http://mongoosejs.com/docs) |
| **openai** | backend/routes/bellaReminders.js for GPT-3.5 calls | [platform.openai.com/docs](http://platform.openai.com/docs) |
| **axios** | Backend (news fetch) & multiple frontend features for API requests | [axios-http.com](http://axios-http.com) |
| **cors** | Express middleware enabling CORS | [github.com/expressjs/cors](http://github.com/expressjs/cors) |
| **dotenv** | Loads environment variables in backend | [github.com/motdotla/dotenv](http://github.com/motdotla/dotenv) |
| **multer** | Handles multipart/form data in routes/rooms.js | [github.com/expressjs/multer](http://github.com/expressjs/multer) |
| **socket.io / socket.io-client** | Backend real‑time API (sockets.js) and frontend fallback in MeetWithFriends.jsx | [socket.io/docs](http://socket.io/docs) |
| **@vapi-ai/web** | Voice assistant integration in Bella.jsx | [docs.vapi.ai/client/web](http://docs.vapi.ai/client/web) |
| **react-router-dom** | Frontend routing | [reactrouter.com](http://reactrouter.com) |
| **react-icons** | Icons used across components | [react-icons.github.io](http://react-icons.github.io) |
| **i18next / react-i18next / i18next-browser-languagedetector** | Language locales via src/shared/i18n.js | [i18next.com](http://i18next.com) |
| **react-qr-barcode-scanner** | QR code scanning in Meals.jsx | [github.com/MadRabbit/react-qr-barcode-scanner](http://github.com/MadRabbit/react-qr-barcode-scanner) |
| **tailwindcss, postcss, autoprefixer** | Styling for the React app | [tailwindcss.com/docs](http://tailwindcss.com/docs)  [postcss.org](http://postcss.org)  [github.com/postcss/autoprefixer](http://github.com/postcss/autoprefixer) |
| **nodemon** | Backend development auto-reload | [nodemon.io](http://nodemon.io) |
| **piper TTS** | Local binary invoked by backend/routes/tts.js | [github.com/rhasspy/piper](http://github.com/rhasspy/piper) |

# **Functions Overview**

# 

## **Backend Functions**

| **File** | **Function** | **Description** |
| --- | --- | --- |
| **backend/api/index.js** | connectWithRetry() | Repeatedly tries to connect to MongoDB with retry logic on failure |
| startServer() | Starts the HTTP/Socket.IO server and handles port conflicts |
| **backend/sockets.js** | cleanupUserFromRoom(userId, roomId) | Removes a user from a room and updates participants; deletes empty rooms |
| Socket events (register, join-room, leave-room, p2p-signal, signal, etc.) | Manage room membership, relay P2P messages, and clean up on disconnect |
| **backend/routes/users.js** | GET / | Return all users from MongoDB |
| POST /addUser | Create a new user document with validation and duplicate check |
| PUT /:id | Update an existing user by ID |
| **backend/routes/contacts.js** | GET /getAll/:userId | Return all contacts for a user ID |
| POST /addContact | Add a new contact record for a user |
| DELETE /deleteContact/:id | Delete a contact by document ID |
| **backend/routes/foods.js** | GET /:barcode | Look up a food item by barcode and return details |
| POST /addFood | Save a new food entry with various nutrition flags |
| **backend/routes/medications.js** | POST /addMedication | Create a medication record for a user |
| PATCH /:id/updateLastTaken | Update a medication’s last taken timestamp |
| **backend/routes/reminders.js** | POST / | Insert a new reminder document |
| GET /:userId | Fetch all reminders for a user ID |
| PUT /:userId/:id | Update a reminder by ID for a user |
| **backend/routes/exercises.js** | GET /elderly-friendly | List exercises flagged as elderly friendly and active |
| POST /populate-sample | Populate the database with sample exercise data |
| DELETE /clear-all | Remove all exercise documents from the collection |
| **backend/routes/news.js** | GET /todays-news | Fetch news articles from the Tagesschau API, mapping them to simplified fields |
| **backend/routes/rooms.js** | POST /create-default | Create a permanent default room and notify clients |
| POST /create | Create a temporary room and add the creator as participant |
| POST /join | Add a participant to an existing room and emit updates via Socket.IO |
| POST /leave | Remove a participant; deletes the room if temporary and empty |
| GET / | List all rooms with participant details included |
| **backend/routes/tts.js** | POST / | Spawn the Piper TTS binary with the proper model and return the WAV file |
| **backend/routes/bellaReminders.js** | POST /addReminder | Manually save a reminder object in MongoDB |
| GET /user/:userId | Retrieve all Bella reminders for a user |
| POST /analyze | Use OpenAI to classify text and optionally store extracted personal information |
| **deno-signaling/server.js** | broadcastToRoom() | Send a JSON message to everyone in a room except an optional user |
| addUserToRoom() / removeUserFromRoom() | Track user membership and inform other participants when users join or leave |
| handleWebSocket() | Handle all WebSocket signaling messages: join, leave, offer, answer, ICE, ping/pong |
| HTTP handlers (/, /health, /stats) | Respond with server info, health, and room statistics with CORS headers |

## 

## **Frontend Functions**

## 

| **File** | **Function** | **Description** |
| --- | --- | --- |
| **src/App.jsx** | fetchJson(url) | Helper to fetch JSON and throw on HTTP errors |
| App() | Root component that loads the first user, manages dark mode, and renders routes |
| **src/components/Header.jsx** | useEffect hooks | Update date/time, obtain geolocation, and fetch weather details |
| **src/components/DenoP2PSignaling.js** | connect() | Open a WebSocket to the Deno signaling server and handle reconnection/ping logic |
| sendOffer/Answer/IceCandidate() | Send WebRTC signaling messages to a specific peer via WebSocket |
| disconnect() | Leave the room and close the WebSocket connection gracefully |
| **src/components/WebRTCManager.js** | initialize() | Create an RTCPeerConnection, add local tracks, and start negotiation if initiator |
| sendP2PMessage() | Send data over the RTC data channel, with fallback to signaling if unavailable |
| handleSignal({signal}) | Process incoming offer, answer, or ICE candidate messages |
|  |  |
| destroy() | Tear down the peer connection, tracks, and timers |
| **src/features/Bella.jsx** | classifyIntent(text) | Lightweight text classifier used on speech transcripts to trigger actions |
| getAssistantId() | Map current i18n language to the proper Vapi assistant ID |
| Call controls (startCall, endCall, toggleCall) | Start/stop the Vapi voice session |
| **src/features/Calendar.jsx** | fetchEvents() | Load reminders for the current user via Axios |
| fetchWeather() | Retrieve a 7‑day weather forecast based on geolocation |
| openNew, openEdit, deleteEvent, saveEvent | Modal handlers for creating, editing, deleting calendar events |
| **src/features/CallContacts.jsx** | toggleSelect(id) | Mark/unmark contacts for bulk deletion |
| handleBulkDelete() | Delete all selected contacts from the backend |
| handleSave() | Add a new contact using form state values |
| **src/features/Exercise.jsx** | fetchExercises() | Get the exercise list (with fallback to HTTP) and populate state |
| filterExercises() | Apply category and difficulty filters to the list |
| speakText(text, id) & stopSpeaking() | Play or stop text‑to‑speech descriptions of exercises |
| populateDatabase() | Send preset sample exercises to the backend API |
| **src/features/Medication.jsx** | markTakenNow(index, id) | Update medication timestamps locally and on the server |
| saveMedication() | POST a new medication entry to the backend |
| askDelete, cancelDelete, confirmDelete | Confirm and remove medication records by ID |
| **src/features/Meals.jsx** | fetchAllMeals() | Load meal data from the API and store it locally |
| fetchByCode(code) | Look up a meal by barcode and speak its description |
| toggleScanner() | Turn the barcode scanner on or off with spoken prompts |
| createFoodDescription(item) | Build a TTS description string for a meal including allergens and additives |
| **src/features/News.jsx** | fetchTodaysNews(regions) | Download news from the backend with retries on error |
| speakText(text, index) / stopSpeaking() | Start or stop audio playback for a news article |
| createNewsDescription(article) | Produce a spoken summary string for an article |
| **src/features/MeetWithFriends.jsx** | createRoom() | POST to backend to create a temporary room and join it |
| joinRoom(name) | Join an existing room, obtain media, and establish P2P connections |
| leaveRoom() | Stop media streams, disconnect peers, and leave signaling rooms |
| toggleAudio() / toggleVideo() | Mute/unmute local tracks and broadcast the state to peers |
| **src/features/SettingsModal.jsx** | changeLanguage(lng) | Switch the UI language and persist the choice in local storage |
| changeUser(e) | Select which user is active by ID from a list |
| toggleAllergen(key) | Add or remove allergen flags in the user profile |
| saveHealth() | Persist updated allergen and diabetic settings to the backend |

# **Setup Overview**

## **Requirements**

- Node.js (current LTS recommended)

- MongoDB instance and connection string  
 *It’s important that MongoDB will have at least one user already defined*

- Deno runtime (for optional signaling server)

- Environment variables: *MONGODB\_URI, OPENAI\_KEY*, optional *PORT*, *VITE\_VAPI\_*\* keys, *TTS\_MODEL\_EN/DE* etc.

## **Setup Walkthrough**

Clone repository:

git clone https://github.com/Leontarin/CareBell

cd CareBell

Backend server:

cd backend

npm install

***#create*** *.****env*** *with* ***MONGODB\_URI*** *and* ***OPENAI\_KEY***

npm run dev

Frontend app:

cd CareBell

npm install

***#create .env*** *with* ***VITE\_VAPI\_PUBLIC\_KEY etc****.*

npm run dev

Deno signaling server:

cd deno-signaling

deno task dev

Ensure MongoDB is running and update src/shared/config.js if API URLs differ.

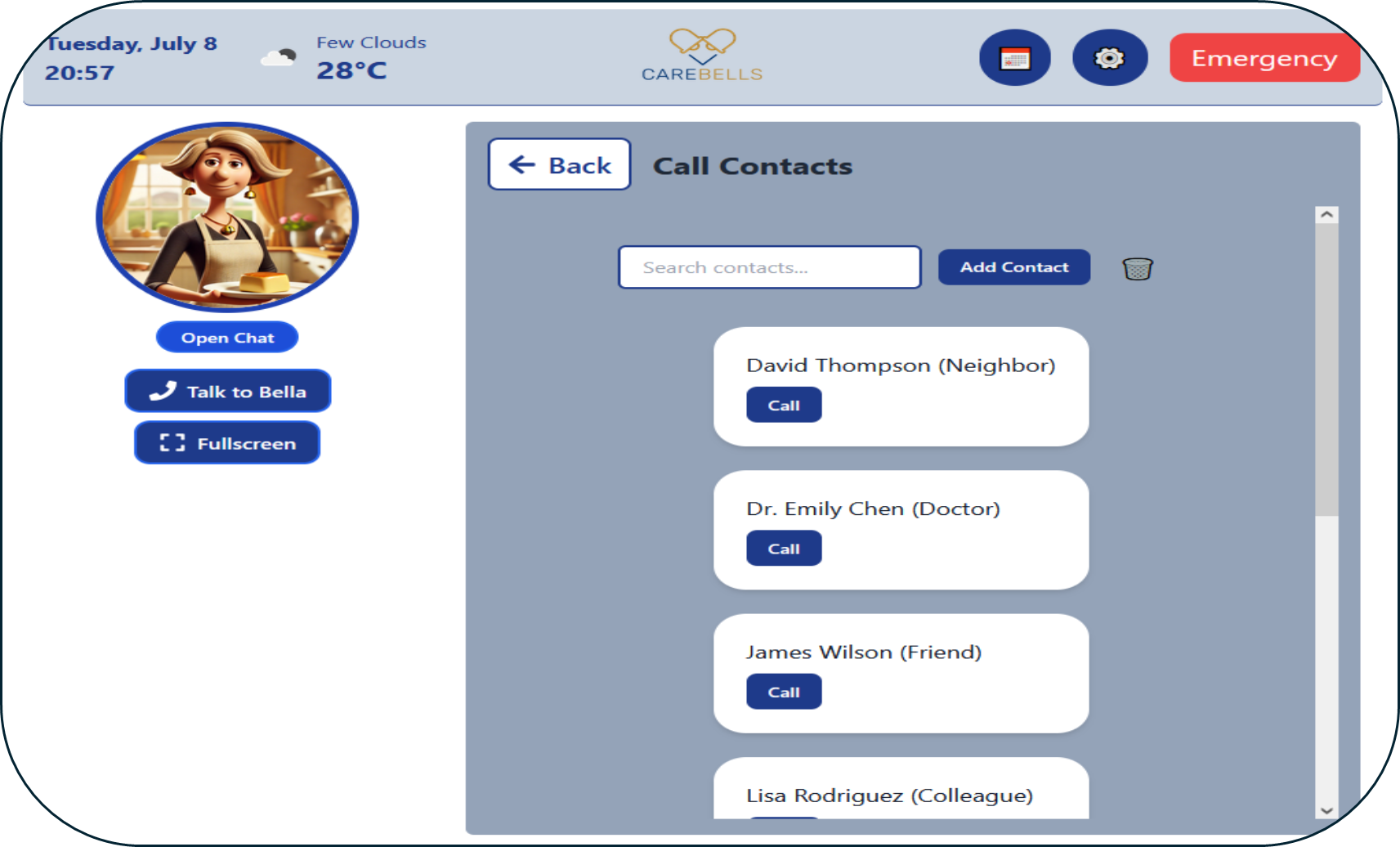
## 

**A README with further details is available inside the GIT REPO**

# CareBells User Guide (תיק למשתמש)

### **1. Home Page Header:**

**Date & Time** (real-time, updates automatically   
 each minute).  
**Weather** (live, based on user’s location).  
**CareBells Logo** (button → returns to Home).  
**Calendar Icon** (button → opens Calendar   
screen).  
**Settings Icon** (button → opens Settings   
modal).  
**Emergency Button** (currently disabled; future   
“Emergency” function).  
Site language is auto-detected from the user’s OS locale (EN, HE, DE or FI; defaults to EN).

**Left Side Chat Controls  
Open Chat** (button → slides in compact Bella chat panel).  
**Talk to Bella** (button → opens full-screen AI chat).  
**Fullscreen** (button → Bella chat in true full-screen overlay).  
  
**Main Menu Right Side Cards**Clicking any card navigates to the relevant feature’s page. 

### **2. Call Contacts**

**Search field** filters your saved contacts in real time (starts empty).

**Add Contact** (button) → opens a form: enter name, phone, role, etc., then Save.

**Trash icon** (button) → enter “delete mode”: select one or more contacts, then confirm deletion.

**Call** (button next to each contact) → launches the device’s native phone dialer to call that person.

### **3. Meet With Friends**

**Enter room name** + **Create Room** (button) → makes a new video room.

**Available Rooms** list:  
Room title “👥 X/10 participants” count Creation timestamp  
**Join Call** (button) → enters that room.

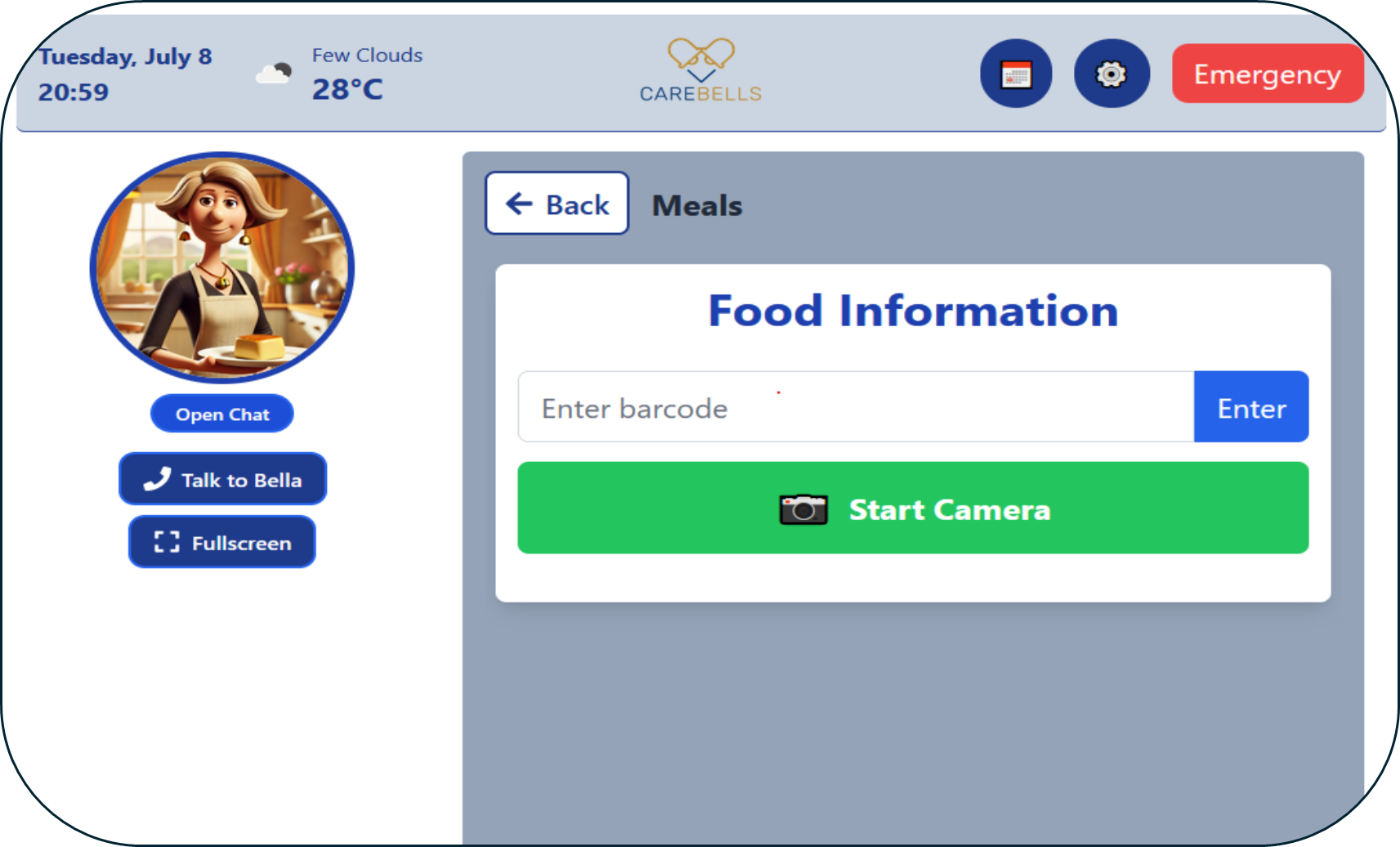
Empty user-created rooms with zero participants are automatically removed.

When a room has participants, a **View Participants** button appears → shows a popup list of everyone inside.

### **4. Medicine**

**Add Medication** (button) → opens a form: Name, Dosage, Frequency, then Save.  
Each medicine card shows:  
 Name (e.g. “Lisinopril”)  
 Dosage (e.g. “10 mg”)  
 Frequency (e.g. “every 12 hours”)  
 Last Taken timestamp  
 Next Due timestamp

**Mark as Taken** (button) → logs the dose,   
updates both timestamps, and disables the button until near the next due time.

**Delete** (button) → asks for confirmation, then removes the medicine record.

### **5. Meals (Food Information)**

**Barcode Entry**: two ways to look up a meal:  
1. Type the barcode number into the “Enter   
 barcode” field + **Enter** button.  
2. **Start Camera** (button) → opens your   
 camera with voice-guided scanning.Once scanned or entered, you’re taken to the   
meal’s detail screen.

### **6. Meals (Example: Poultry Sausage)**

Shows:

**Image** of the meal  
**Name** (“Poultry Sausage”)  
**Barcode** (“2000”)  
**Description** (“kohlrabi vegetables and   
 mashed potatoes”)  
**Diabetic friendly:** ✔️ or ❌ (based on   
 your stored dietary profile)

A yellow warning banner for any   
allergen that the meal contains and you   
are allergic to. (if you are not allergic to any of the ingredients in the meal it will not show the warning banner)

**Read** (button) → uses text-to-speech to speak the meal name, ingredients and allergen info aloud.  
**Stop reading** (button) → immediately halts any ongoing TTS playback.

### **7. News**

**Select Regions** (dropdown) → choose one or   
more regions to filter your news feed.

Displays “**X articles found**” and lets you scroll  
 indefinitely through the list.

Each article shows:  
 Headline + image  
 Source name & timestamp  
**Read** (button) → reads the article text aloud.  
**Stop reading** (button) → immediately   
halts any ongoing TTS playback.

### **8. Exercise Library**

**Category** filter (dropdown): All | Strength |   
 Flexibility | Cardio | Balance  
**Difficulty** filter (dropdown): All | Easy |   
 Medium | Hard  
Exercise cards display:  
 Image thumbnail  
 Title (e.g. “Chair Squats”)  
 “Category: … | It lasts for: X min”  
 Difficulty badge (e.g. “Easy”)

Clicking **View Details** on a card opens its full   
instructions screen.

### **9. Exercise Detail (Chair Squats)**

**Title & a gif** at top. The gif shows how to   
practice each exercise.

**Description:** a short overview.

**Instructions:** numbered steps.

**Benefits:** bulleted list of outcomes.

**Precautions:** red bullets (e.g. “Hold onto chair   
 for balance”).

**Modifications:** green bullets (e.g. “Partial squats only”).

**Read Instructions** (button) → reads the instructions aloud.

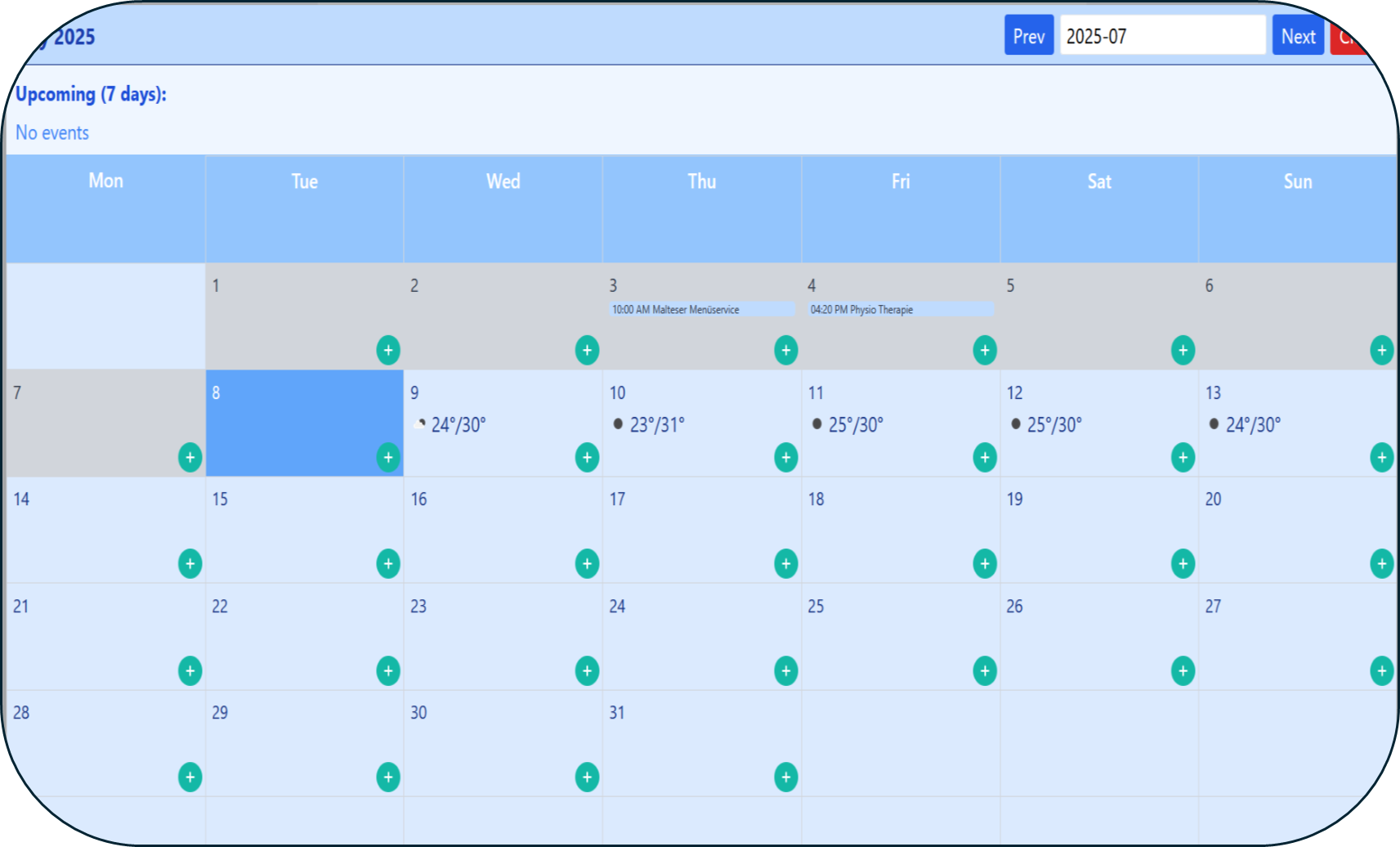
### 

### 

### 

### 

### **10. Talk to Bella**

**VoiceChat interface** with AI assistant “Bella”:  
Voice chat bubbles display both Bella’s   
replies and your voice messages.  
Global chat controls (same as Home Page sidebar):  
**Open Chat** → opens the chat panel. **Close Chat** → hides the chat panel.  
**Talk to Bella** → opens full-screen voice chat and activateS “Bella”..  
**Fullscreen** → opens the chat in a full-screen mode.

### **11. Calendar Prev / Next** (buttons) → navigate to the previous or next month. **Month view**: weekdays across top, date cells below. **Upcoming (7 days):** lists any events you’ve added for the coming week. **+** (button in each date cell) → opens a modal popup to enter event title, date & time. Tapping each day will open a grid of hours of that day and will show if there's any events on that day. Also, you can tap on each hour and it will work the same as the “**+**” button.

### **12. Settings**

**General** tab:  
**Text Size:** slider between small and large.  
**Volume:** slider (currently non-functional).  
**Speaking Speed:** slider (currently non-functional).  
**Language:** dropdown (English, Hebrew, German, Finnish).  
**Change User:** dropdown (development only; will be removed in a final product).  
**Dark Mode:** toggle switch → switches the entire UI into dark theme.  
**Health** tab (development only): planned place to set user allergy profiles..

**קישור לגיט:** <https://github.com/Leontarin/CareBell>

**קישור לאתר:** <https://carebell.vercel.app>