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