Front-End File Explanations

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
`npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf`

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:

```
npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
`npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf`

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:

```
npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:

```
npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
`npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf`

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:

```
npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all CSS, JavaScript and renders three main views (Dashboard, History, Profile) through hidden/visible sections. It also links Chart.js, Bootstrap/Tailwind (if added) and contains placeholders (`<canvas>`, forms, tables) that the JS files populate at runtime.

---

css/styles.css

Custom styles that complement any framework.
• Defines colour palette variables, button states and form tweaks.
• Adds layout specifics (e.g., `.sidebar`, `.card`, `.hero`).
• Media queries ensure mobile responsiveness.

---

js/api.js

Wraps **all HTTP calls** to the Express backend.

- `getAuthHeader()` injects the JWT from `localStorage`.
- CRUD helpers (`fetchGlucose`, `addGlucose`, `login`, `register`) use `fetch` and return raw JSON to callers.
- Centralises error handling: a `handleResponse` function throws on non-2xx so UI code stays clean.

---

js/app.js

Acts as entry point after DOM ready.

- Determines which page is active via URL path and initialises the matching module (dashboard/history/profile).
- Registers global event listeners (logout, nav clicks, theme toggle).
- Provides small utility functions like `showToast(message, type)`.

---

js/auth.js

Handles sign-in/sign-up UI.

- Attaches submit handlers to login/register forms.
- On success, stores token + user data in `localStorage` then redirects to `/dashboard`.
- Includes `requireAuth()` guard to prevent unauthorised view access.

---

js/dashboard.js

Core glucose entry & visualisation logic.

- `loadStats()` pulls latest readings to compute average, min, max and updates KPI cards.
- `renderChart()` draws the time-series line chart with Chart.js and the annotation plugin for target range shading.
- `handleGlucoseEntry()` posts new value, refreshes chart and, if out-of-range, triggers **VoiceModule.speak** plus optional SMS/e-mail depending on user prefs.

---

js/history.js

Renders paginated table of past glucose readings.

- Calls `api.fetchGlucose({page,limit})` then builds rows dynamically.
- Allows deletion/edit via icons—sends PUT/DELETE then updates table.
- Offers CSV export by converting the current table into comma-separated lines and triggering a download.

---

js/profile.js

User settings management.

- Loads current profile (name, age, target range, guardian contact).
- Saves updates via `api.updateUser()` and shows a success toast.
- Toggles notification preferences (voice, SMS, e-mail) and writes them to user document.

---

js/voice.js

Lightweight wrapper around the Web Speech API.

- `VoiceModule.speak(text)` selects an English voice, cancels any ongoing utterance, and speaks the supplied message.
- Dispatches a custom `voicemodule-ready` event once voice list is loaded so dashboard can wait before speaking.

How to Export This Document as PDF

1. Open this file in VS Code or any markdown viewer.
2. Use the built-in **Markdown: Print to PDF** command (or another extension) **OR** copy-paste into an online converter.

> Tip: if you prefer an automated approach, install **markdown-pdf** (`npm i -D markdown-pdf`) and run:
>
> ```
> npx markdown-pdf docs/frontend_explanations.md -o docs/frontend_explanations.pdf
> ```

Generated with ❤ to help you quickly understand and document your front-end architecture.

This document describes the purpose and key functionality of every file under `src/frontend`.

---

index.html

The single-page shell that loads all