ActivePoints Codebase Documentation

GDP Group 9

Ayaz Baig (mab1g21@soton.ac.uk)
Leah O'Loughlin (lo3g20@soton.ac.uk)
Kagethan Thayarathan (kt3g21@soton.ac.uk)
Alyssa Sieradzki (ams2g21@soton.ac.uk)

January 2025

Contents

1 Codebase Changes

1.1 Dashboard Updates

This section covers the updates to the clinician dashboard which includes the addition of the authentication functionality, patient assignments and blood glucose data visualisations.

1.1.1 React Frontend

Authentication
The main entry point, `App.js`, has now been wrapped with an authentication context provider, found in `src/scenes/Authentication/AuthContext.js`. This stores a local browser boolean variable `isLoggedIn` to flag is a clinician has already logged in. This use of browser storage allows for a persistent login to the dashboard. This context provider also has methods to login and logout, which modify the respective flags and authentication variables where necessary.

The Home page and the user Dashboard page have been wrapped in a private route. This can be found in `src/utility/privateRoute.js`, which redirects users to the default `'/'` route if they are not logged in. The default `'/'` route now navigates to the AuthPage which handles the login functionality.

The `AuthPage` found in `scenes/Authentication/Authentication.js`. This page handles the login form, and stores variables in the browser storage for the clinician and institution details, which are used to personalise the main dashboard home page.

Adding patients
The functionality to add patients is done using the `AddPatientButton` found in `scenes/PatSelect.js`. Pressing this button shows the `AddPatientPopup`, which the clinician can use to enter a user's email address to add them as a patient.

Patient list
The functionality to load patients has been modified to load a clinician's assigned patients, rather than a pre-defined list of users. This is done in the `refreshUserList` method, also in `scenes/PatSelect.js`. The `CardRow` component builds a `PatientCard` for each patient returned from the method.

Glucose Graph
The `Dashboard` component in `scenes/Dashboard.js` has been updated to add an extra toggle for glucose data, and also has modified the existing functions to include glucose data. This works similar to how the other health metrics are retrieved from the Express backend endpoints.

The glucose graph changes can be found in `components/Graph/Graph.js`, which draws a line graph to show glucose progression with a horizontal target line to show the glucose target value.

1.1.2 Express Server Backend

Database Connection
The `db.js` has been updated to create the connection pool using the `DATABASE_URL` environment variable.

Clinician authentication
New endpoints have been made to handle authentication. The `institutions` endpoints retrieves a list of current institutions from the database. The `signup` and `login` endpoints handle signup and login functionality, respectively, by assigning and validating a clinician using their first name, last name and institution.

Adding patients
The `clinician-user-ids` endpoint returns a list of user IDs which have been assigned to a clinician. The `clinician-user-details` endpoint returns the details of the users which have been assigned to a clinician. Both endpoints take the clinician ID as a parameter in the request's payload body. The `search-user` endpoint returns the basic details of a user if they exist given an input email. The `add-user` endpoint assigns a specified user to the specified clinician based on the input user ID and clinician ID.

The `daily_data` endpoint and the `fetchAPData` method have been modified to include the glucose values from the Libre table.

1.2 Strapi Updates

The main Strapi update required updating the Strapi dependency plugins from 4.9.1 to 4.25.15.

New collections
New collections were added for Clinician, Institution, Libre and Libre Daily Data using the Strapi admin panel. The Users collection was modified to include relationships with the Clinician and Libre collections. An API token was also generated using the admin panel, which is added in the Express server environment variables.

Cron Tasks
New methods were added to the `config/cron-tasks.js` file to implement the Libre cron jobs. These are `getLibrePreviousDayData` and `getLibreData`, which modify the Libre Daily Data collection and create new entries for glucose values at new timestamps. These methods are run similar to the Fitbit and Garmin cron jobs, with the `getLibrePreviousDayData` running at 5 minutes past midnight each day, and `getLibreData` running every 15 minutes.

Push Notifications
A new route to handle saving FCM tokens was added to the Users collection seen in **extensions/user-permissions/strapi-server.js**. These FCM tokens are used to receive push notifications from Firebase.

1.3 Mobile App Updates

1.3.1 Libre Integration

Connect Libre Frontend Changes
A new Libre component has been created in `src/components/Libre/Libre.js`, which handles pressing the new 'Connect with Libre' button. This button has been added to the `ConnectWatch` container. When the button is pressed, it displays the `EnterLibreViewInfo` form which the user can then enter their LibreView info to connect their device.

Libre API Connections
A new hook has been added in `hooks/useConnectLibre.js`, to match the integration flow of the Fitbit and Garmin devices. This hook includes methods to handle each stage of the Libre integration flow and make the required requests to the `User` Redux-slicer in `redux/slicers/user`. The slicer includes methods to handle integrating with Libre, disconnecting the device and getting glucose data, and also includes additional state variables to track if a Libre has been connected and store Libre device data. This links with the `User` Redux-saga in `redux/sagas/user`. The `User` saga connects to the newly made `VitalClient` in `config/VitalClient.js`, which includes methods that make the connection requests to the Vital API.

Additionally, the `WebService` has been modified in `config/WebService.js` which includes request payloads and configurations to the Strapi API framework, allowing for creating, updating and deleting Libre devices for a user.

Glucose Widget
The home page has been updated in `containers/Home/index.js`, to now include a new widget to display glucose data from the Libre device. This also makes a call to the `fetchLibreData` method in `useConnectLibre` to retrieve the latest glucose reading via Vital API.

1.3.2 Firebase Notifications Configuration

Firebase Connection
To create the link to the Firebase apps, the `google-service.json` and `GoogleService-Info.plist` files were downloaded from the Firebase console and placed in the `android/app` and `ios/ActivePoints` directories respectively. The permissons for Android and iOS to enable this connection was configured in the `android/app/src/main/AndroidManifest.xml` file for Android, and `ios/ActivePoints/Info.plist` file for iOS.

Notification Handling
The `src/index.js` file was modified to ask for notification permissions on app startup, and create the background message handler to display notifications while the app is in the background. The helper files `ForegroundNotification.js`, `PushNotification.js`, `firebaseHelper.js` and `notifications.js` were modified to correctly handle incoming notifications from Firebase Cloud Messaging, while the app is in the foreground.

2 Operational Guide

2.1 Dashboard Operations

The clinician dashboard can be accessed on `https://d10aelppdvczc2.cloudfront.net/`.

2.1.1 Adding Institutions & Clinicians

A new institution can be added via the Strapi admin panel in the Institution collection, which only requires a unique name field. The clinicians for this institution can also be assigned when adding this institution.

A new clinician can be added also via the Strapi admin planel in the Clinician collection, which requires a first name and last name. The clinician can be assigned to an institution when creating this clinician, via the dropdown input.

2.1.2 Assigning patients

A user can be assigned as a patient for a clinician on the clinician dashboard using the Add Patient button. This requires entering the user's email address, and if a user is found with this email, the option is shown to confirm adding this user.

Alternatively, a user can be assigned to a clinician via the Strapi admin panel. This can be done either through the clinician's entry field in the Clinician collection, or through the user's entry field in the User collection.

2.2 Mobile App Operations

2.2.1 Linking Libre Device

To link a Libre device, a LibreLink app account is necessary to first pair the Libre device with a LibreView account which the Vital API uses to connect on the ActivePoints mobile app. Once the LinkLink app is setup, the Libre device can be paired by positioning the phone to the device until it vibrates. This confirms the device has been uploaded and connected to LibreLink which makes it visible on LibreView.

After these initial steps, navigate to 'Connect your device' and press 'Connect FreeStyle Libre' to show the Libre connection form. Enter the LibreView account credentials which will prompt the two-factor authentication code to confirm access. If all credentials and codes are valid, the connection will be established, and the glucose widget should now be visible on the home page with the latest glucose reading.