PPDV – final project report

No.	Name	Surname	Student ID
1.	Daria	Danieluk	319031
2.	Weronika	Zbierowska	319130

The goal of the final project was to create a web-based application written in Python using the Plotly Dash framework for data visualization. The application presents a graphical visualisation of measurements from a device for monitoring walking habits and patterns of elderly and disabled patients. The data is fetched from an API hosted on a Faculty of Electrical Engineering server (accessible after connecting to the EE VPN). There are 6 patients for whom the measurements are available.

Design

The application was designed for physiotherapists. We intended for the design to be straight-forward and understandable. The physiotherapist can follow the last 10 minutes of walking of a set of line graphs (1 for each sensor), the live measurements with visualised pressure values of the sensors on a feet diagram and the history of anomalies for each patient on similar line graphs as in the live view. The patient personal information is also display to include context for the physiotherapist. Zooming in on the line graphs is supported.

We wanted to keep the design simplistic to maximise the perceived data-ink-ratio.

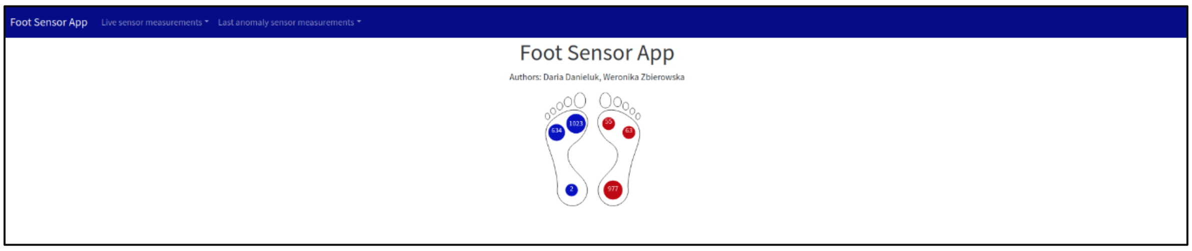
Views

The application consists of 3 views:

- homepage
- live sensor measurements
- last anomaly sensor measurements

Homepage

The homepage is a simple landing page consisting of the navigation bar, the app name, names of the authors and an example of a custom component used in the application. The navigation bar consists of 2 dropdown menus directing the user to the 2 possible views of the application. Each dropdown menu has a list of available patients (1-6).


Foot Sensor App
Live sensor measurements ▾
Last anomaly sensor measurements ▾
634
1023
2
55
63
977
Foot Sensor App

Authors: Daria Danieluk, Weronika Zbierowska

Home page view

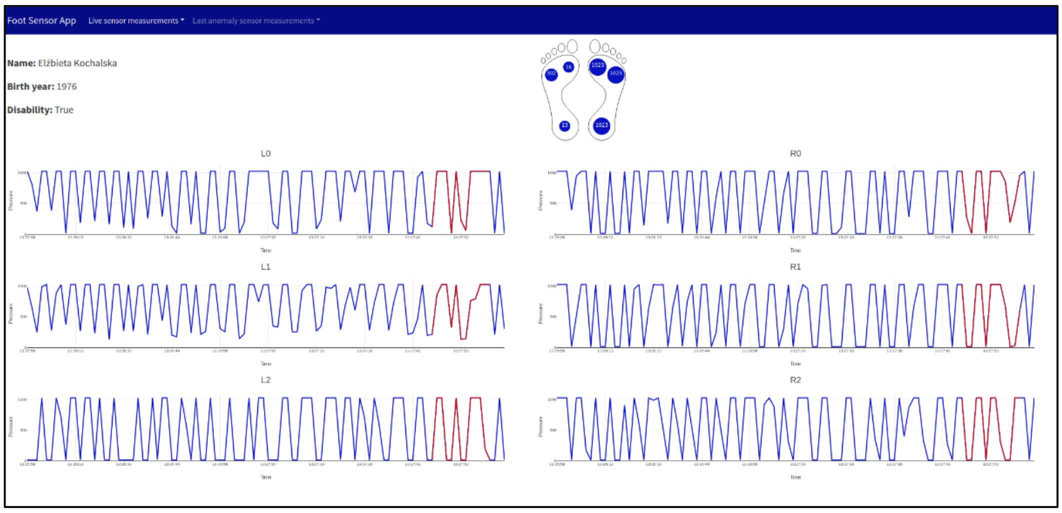
Live sensor measurements

This view allows the user to see real-time visualisations of the process of walking. It consists of 3 main visual components:

- patient personal information:
  - full name
  - birth year
  - disability status
- feet diagram with 3 sensors on each foot:
  - size of the sensors changes based on the pressure measurement
  - colour of the sensor changes based on anomaly detection:
    - blue – no anomaly detected
    - red – anomaly detected
  - placement of the sensors:

No.	Name	Placement
1.	L0	left foot front
2.	L1	left foot middle
3.	L2	left foot back
4.	R0	right foot front
5.	R1	right foot middle
6.	R2	right foot back

- a set of 6 graphs representing the process of walking:
  - each graph represents one sensor
  - graphs on the left side visualise the left foot and graphs on the right side visualise the right foot
  - Y axis - possible pressure values in the range [0, 1023]
  - X axis - timestamps of last 10 minutes of walking (or since the start of the measuring process if 10 minutes have elapsed yet), max. 10 tick values displayed
  - the colour of the trace depends on the detection of an anomaly:
    - blue – no anomaly detected
    - red – anomaly detected
  - zooming in is possible



Name: Elżbieta Kochalska

Birth year: 1976

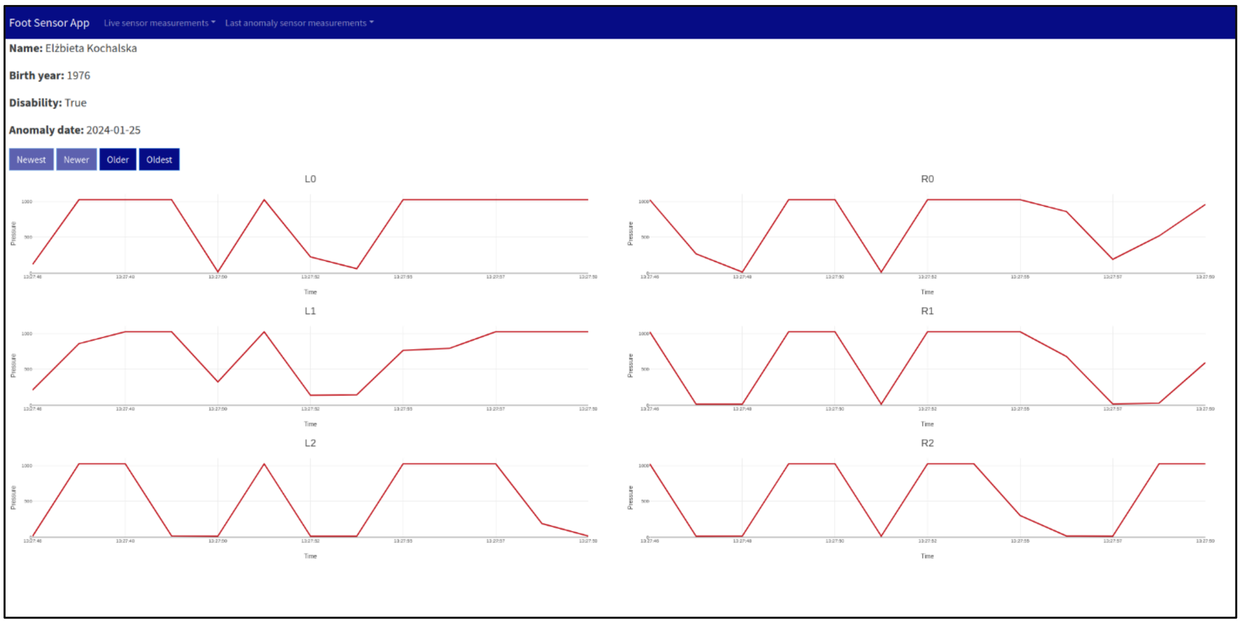
Disability: True

Live sensor measurements

Last anomaly sensor measurements

This view allows the user to see the historical traces that have been detected as an anomaly. It consists of 3 main visual components:

- patient personal information:
  - full name
  - birth year
  - disability status
  - shown anomaly detection date
- a set of 4 buttons:
  - Newest – go to the most recently detected anomaly trace
  - Newer – go to the more recently detected anomaly trace than the currently viewed one
  - Older – go to the previously detected anomaly trace from the currently viewed one
  - Oldest – go to the first detected anomaly trace
- a set of 6 graphs representing the process of walking during the detected anomaly:
  - each graph represents one sensor
  - graphs on the left side visualise the left foot and graphs on the right side visualise the right foot
  - possible pressure values in the range [0, 1023]
  - traces are red to signify an anomaly



Name: Elżbieta Kochalska

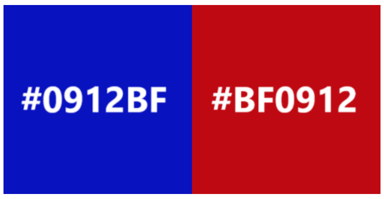
Birth year: 1976

Disability: True

Anomaly date: 2024-01-25

Last anomaly sensor measurements

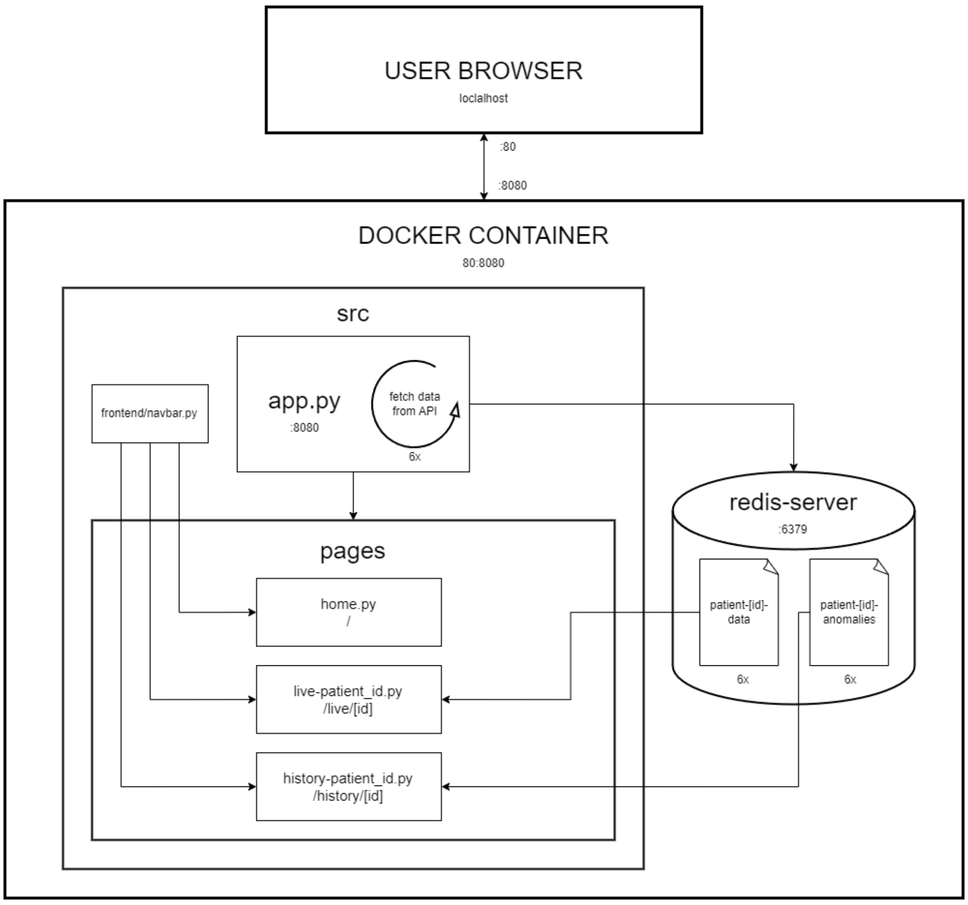
Colours & styles



Colour palette

We have decided to use an external style sheet “Cosmo” from the “dash_bootstrap_components” module. This helped us achieve a consistent look for the application. We wanted to keep the theme of the application in blue shades that are associated with the medical industry. On the graphs and the feet diagram we have decided to use blue (#O012BF) for non-anomaly data, because it’s considered neutral, and red (#BF0912) for anomaly data, because human naturally associate red with danger and need for attention. We also took into consideration the fact, that this colour combination will be easy to differentiate by colourblind users.

Architecture



The application is containerised using Docker. The container has 1 port mapping *80:8080*, which means that the main application (run at port 8080) is available on the host’s port 80 at the address *localhost*.

The application consists of 2 parts: the Dash app and the Redis database. Data fetching is done in the main app using multithreading. Each thread is responsible for fetching data from an API for 1 patient every 1 second and saving it to the *patient-[id]-data* list in the Redis DB. Anomaly traces are additionally stored in another list *patient-[id]-anomaly* in the Redis DB.

Pages are implemented using the Dash feature “use_pages” which is meant for easier managing of multi-page applications. They are stored in the *pages* directory and each page, representing 1 of the views, is registered in a separate file.

Module “live-patient_id.py”

On page load all data from a Redis DB list *patient-[id]-data* is fetched for a given patient. It is stored separately for each user in a “dcc.Store” component to allow for a smooth multi-user experience. Then a new data record is fetched every 1 second. This allows for an update of the graphs and the feet diagram in real time. Data older than 10 minutes is deleted and not displayed to the user.

Module “history-patient_id.py”

On page load all data from a Redis DB list *patient-[id]-anomalies* is fetched for a given patient and the newest anomaly trace is displayed on the graphs. When user presses a button, a different trace is shown (newer or older). Appropriate buttons are disabled when there is no older or newer trace to display. An id of the currently displayed trace is stored separately for each user in a “dcc.Store” component to allow for a smooth multi-user experience.

Custom component

The custom feet diagram component has been created using the *dash component boilerplate*. The code responsible for displaying the component has been written in React. After building, the component has been packaged into a tarball *feet_sensors-0.0.1.tar.gz* using the provided *setup.py* script. The component was made available in the main project code by installing it using pip.

Running the project

1. Connect to the EE VPN.
2. Build docker container:

   ```
   $ docker build -t foot-sensor-app .
   ```

3. Run docker container:

   ```
   $ docker run --name foot-sensor-app -d -p 80:8080 foot-sensor-app
   ```

4. Access the app at “localhost”.
5. To stop the container:

   ```
   $ docker stop foot-sensor-app
   ```

6. To restart the container:

   ```
   $ docker start foot-sensor-app
   ```