# FallDec Dashboard
## User manual

Konrad Skarżyński, Inga Maziarz

January 2024

## Contents

# 1 Introduction

The *FallDec Dashboard* serves as a comprehensive and user-friendly web application tailored for individuals utilizing the *FallDec* feet pressure measurement device. This user manual is designed to provide an in-depth understanding of the installation, functionality and usage of the dashboard, enabling users to effectively monitor and interpret data collected by the device.

# 2 Getting started

## Prerequisites:

To run the program, it is necessary to install the Python 3 interpreter on your device. Additionally, you need to have the Python libraries specified in the *requirements.txt* file installed. Your device must be connected to the EE VPN as the program gathers data from a website accessible only when connected to the network of EE faculty. Automatic detection of the operating system and valid installation files is performed on `https://vpn.ee.pw.edu.pl`.

## Installation:

You can download the program from GitHub by accessing the repository at the following link: `https://github.com/imaziarz/webpage`. Unzip the archive into a selected directory (make sure to set your working directory accordingly). Afterward, execute the following command in your terminal:

```
python widget/webpage.py
```

After executing command above, a redirection to the webpage should appear. By clicking it (or alternatively, copying it into your browser), the application will open.

# 3 Architecture

## Database Creation:

The script creates an SQLite database named patients.db with two tables: patients and anomalies.

## Data Fetching and Storage:

The fetch and store data function continuously retrieves patient data from an external API, processes it, and stores it in the patients table of the database. It uses the requests library to make API calls and the pandas library to manipulate and store data in the database.

## Anomaly Detection and Copying:

The copy anomalies function runs in the background, identifying patients with anomalies and copying corresponding data to the anomalies table.

## Web Application with Dash:

The Dash framework is used to create a web application with a user interface. The main layout includes patient information, a patient list, a real-time graph, a sensor checklist, and a custom widget (probably for displaying sensor values). The web application updates every second (dcc.Interval(id='interval', interval=1 * 1000, n intervals=0)).

## Callbacks:

Dash callbacks are used to update the web application dynamically based on user interactions and data changes. Callbacks are triggered by events such as selecting a patient, changing sensor preferences, or the defined time interval.

### Graphical Representation:

The real-time graph is plotted using the plotly library, showing sensor values over time for selected sensors. Anomalies are marked in red.

### Multithreading:

Multithreading is employed to run the data fetching and anomaly detection functions concurrently in the background, ensuring that the web application remains responsive.

### Widget and Patient List:

The web interface includes a custom widget (widget.Widget) and a patient list table displaying relevant patient information.

### Server Setup:

The Flask server is initialized with the Flask( name ) line, and the Dash application is created with app = Dash( name , server=server).

### Execution:

The script checks if it's the main module and, if so, starts the Flask server to run the web application.

## 4 Usage

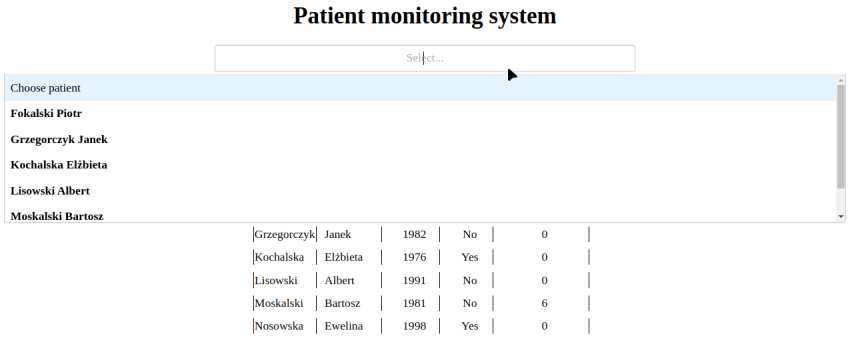


Figure 1: Choosing patient from list

To display information on the dashboard, you should first select a patient of interest from the list (1). You can find basic patient data in the table above. After selecting a patient, a chart and a widget representing foot pressure (collected from sensors over the last 10 minutes) will be displayed (2). You can include specific sensors on the chart by selecting the checkboxes below it. On the right side, a legend will assist you in interpreting the results. Points identified as anomalies will appear in red. Below the chart, a widget displays real-time changes in foot points, with the point size dependent on the sensor-collected value. At any moment, you can switch to monitoring a different patient.

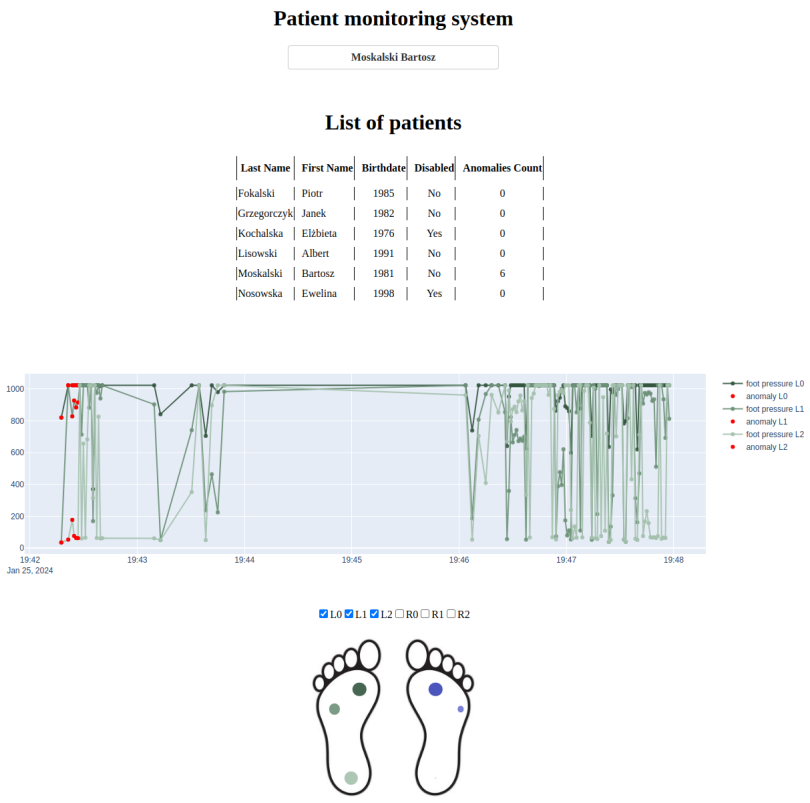


Figure 2: Dashboard layout