DAIsy
Developing AI ecosystems improving diagnosis and care of mental diseases
DAIsy
24 partners
5 countries
ITEA Call 2021
Smart health

DAIsy – Developing AI ecosystems improving diagnosis and care of mental diseases

ITEA 4 – 21016

Work package 5 (WP5)

Platform Development

Deliverable 5.2. Visualizations & Monitoring

Document type	: Deliverable
Document version	: No. 1
Document Preparation Date	: Feb 2024
Classification	: Confidential
Due Date	: Dec 2025

HISTORY

Document version #	Date	Remarks
1	Feb 2024	Initial version
2	Feb 2026	Final version

Table of Contents

1 Introduction

The implementation of monitoring and visualization components is a critical aspect of data analytics. For system developers and clinicians, these components not only facilitate the real-time collection and aggregation of data from diverse sources such as sensors, databases, and external APIs but also enable the effective interpretation of this data through intuitive dashboards and visual representations. By focusing on a user-friendly design, the threshold for interaction is minimized, making the system accessible to individuals regardless of their technical background. This deliverable describes the tools and components that have been created and adapted for this purpose within the framework of the Daisy project.

The various security mechanisms of the individual components described are discussed in detail in Deliverable 5.4, where the protective measures and security architecture are comprehensively documented.

2 Description of the method of integration into an existing system

2.1 Integration of Python-based LSL components with GUI elements

The integration of Python-based components into the DAIsy system follows a standardized approach that ensures consistent data flow, low-latency synchronization, and a unified graphical user interface. Each component developed in Python utilizes the Lab Streaming Layer (LSL) protocol for real-time data exchange (Kothe et al., 2025). This enables seamless interoperability between different signal sources, such as EEG and fNIRS devices, and processing or visualization modules within DAIsy. The general method is to configure each component as an LSL node, where it acts as both a data receiver (input) and a data producer (output). This structure allows the component to process physiological data streams, perform calculations, and return derived features or feedback variables to the system. This real-time communication model ensures the integration of new components without modifying existing components, thus maintaining DAIsy's modular and scalable design philosophy.

DAIsy supports the embedding of GUI components developed with PySide6 or QtPy directly into the main application framework. Each GUI module is implemented as a dockable widget that can be dynamically loaded or unloaded according to user needs. This approach enables flexible visualization and control across multiple experimental setups.

The component can be deployed in two modes:

- Embedded mode: The component is imported into the DAIsy environment as a Python module. Its main widget is instantiated and registered as part of the main GUI, while LSL inlet and outlet configuration is handled via the DAIsy Settings Manager.
- Standalone mode: The component runs as an independent application that connects to existing DAIsy LSL streams. In this mode, it can perform preprocessing, visualization, and feedback generation while maintaining full compatibility with the DAIsy ecosystem.

To ensure reproducibility, the deployment process is based on a virtual Python or Conda environment with the specified dependencies installed from a requirements file. Cross-platform compatibility is ensured by multi-OS support for Qt and Python, allowing the same component to run without modification on Windows, macOS, and Linux systems. This standardized integration method provides a clear framework for adding new Python-based modules to DAIsy and supports both GUI and headless configurations. It ensures that data exchange, synchronization, and control interfaces follow consistent conventions. This reduces redundancy in component-level documentation and simplifies long-term system maintenance.

3 Therapy assistant – Therapist Dashboard

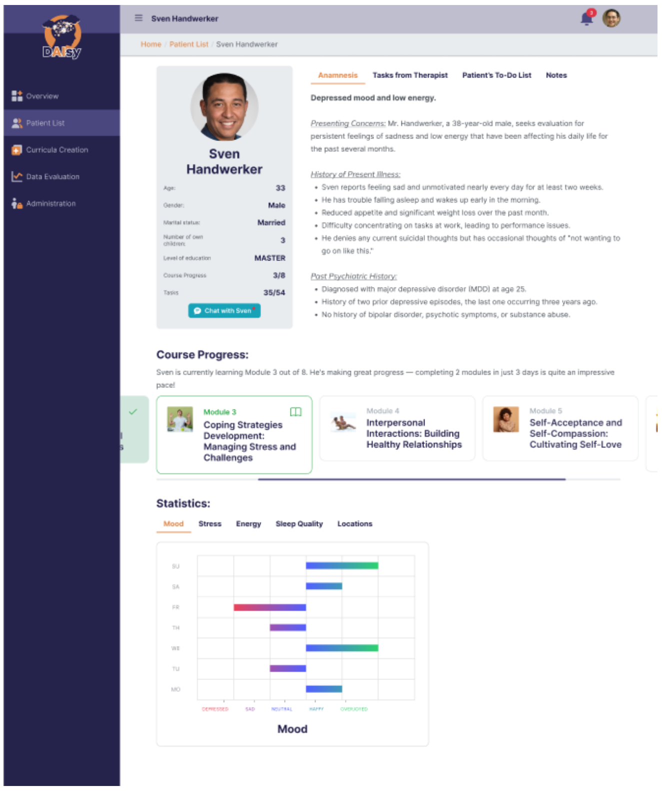
The therapist dashboard serves as a central platform where therapists can not only access their patients' health data, but also manage informational materials and content related to therapy-accompanying work. In addition, it enables continuous monitoring of patients' progress and behavior between therapy sessions. For example, activity patterns, exercise progress, or feedback from digital diaries can be recorded and evaluated in order to tailor therapy to individual needs and respond early to changes in patient behavior.

3.1 Type of data processed

The measurement data from the patient app is displayed and includes both system-generated and patient-reported information. This includes metrics such as step count and app usage, as well as data entered by the patient such as PHQ-9 test results and mood information. This comprehensive data set is made available to the therapist via the therapy assistant dashboard. Here, the therapist can compare and analyze the data to identify correlations and dependencies between different aspects of the patient's health and behavior. In addition, AI-supported progress predictions are generated from the measurement data and can be displayed in the dashboard. These predictions help the therapist to identify possible developments at an early stage and to design the therapy accordingly in a forward-looking manner. Furthermore, patients' curricula can be displayed and customized in the dashboard so that the content taught is optimally tailored to current needs and therapy progress. In addition, information from hospital management can be integrated and displayed, giving therapists a comprehensive overview of patient-relevant organizational and medical data.

To support clinical interpretation, the dashboard also highlights data quality and completeness indicators (e.g., missing days, sensor coverage, or delayed synchronization). This enables clinicians to distinguish genuine clinical change from data sparsity and to interpret trends in a robust manner during follow-up.

3.2 **Opportunities for interaction**



Sven Handwerker

Age:	33
Gender:	Male
Marital status:	Married
Number of own children:	3
Level of education	MASTER
Course Progress	3/8
Tasks	35/54

Depressed mood and low energy.

Presenting Concerns: Mr. Handwerker, a 38-year-old male, seeks evaluation for persistent feelings of sadness and low energy that have been affecting his daily life for the past several months.

History of Present Illness:

- Sven reports feeling sad and unmotivated nearly every day for at least two weeks.
- He has trouble falling asleep and wakes up early in the morning.
- Reduced appetite and significant weight loss over the past month.
- Difficulty concentrating on tasks at work, leading to performance issues.
- He denies any current suicidal thoughts but has occasional thoughts of "not wanting to go on like this."

Past Psychiatric History:

- Diagnosed with major depressive disorder (MDD) at age 25.
- History of two prior depressive episodes, the last one occurring three years ago.
- No history of bipolar disorder, psychotic symptoms, or substance abuse.

Course Progress:

Sven is currently learning Module 3 out of 8. He's making great progress — completing 2 modules in just 3 days is quite an impressive pace!

Module 3
Coping Strategies Development: Managing Stress and Challenges

Module 4
Interpersonal Interactions: Building Healthy Relationships

Module 5
Self-Acceptance and Self-Compassion: Cultivating Self-Love

Statistics:

Mood · Stress · Energy · Sleep Quality · Locations

The data analysis in the therapist dashboard presents information from the patient app, which has been normalized to account for hardware differences between patients. In certain cases, such as determining the distance traveled each day, the patient app aggregates data, which is then displayed in the dashboard. To facilitate comparison and analysis, all data is presented as time series, allowing therapists to correlate different data points over a specific period of time. This comprehensive view helps identify trends and correlations and improve the therapy process.

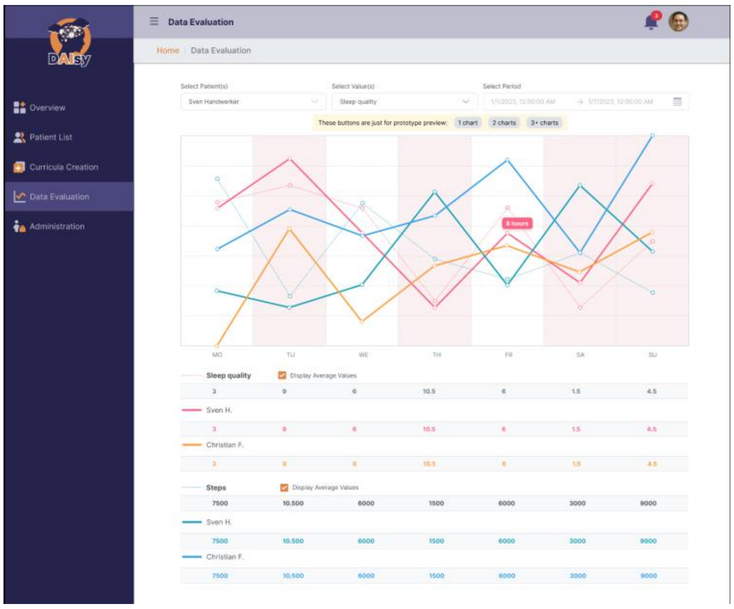
In addition to the measured data, the dashboard also offers the option of customizing teaching content and arranging it in an order that is appropriate for the individual patient. This allows learning and exercise materials to be tailored specifically to the patient's needs and current progress in therapy. Furthermore, the system allows continuous monitoring of patient progress between sessions, so that therapists can track behavior, engagement, and learning outcomes at any time.

There is also the option of assigning specific tasks to the patient—for example, “Take a 10-minute walk once a day.” These tasks can be one-time or recurring and are automatically entered into the patient app. The therapist receives an overview in the dashboard of which of these tasks have already been completed and which tasks the patient may have set for themselves. This information can then be specifically addressed in therapy sessions and integrated into further planning.

In addition, anomaly flags can be presented for abrupt changes in key time series (e.g., sleep disruption, activity drop, or mood deterioration), enabling early review by clinicians. These flags are intended to support monitoring and prioritization rather than automated clinical decisions.

3.3 Interfaces

When analyzing data, therapists can display any combination of a patient's measurement data, all of which is presented in the form of time series to facilitate the identification of relationships between different data points. In addition, therapists can use data from other patients as comparative values. This feature improves the analysis process and provides more comprehensive insights into the patient's health and behavior patterns. The user interfaces are deliberately not designed for data scientists, but are kept simple and clear so that therapists can quickly gain a clear overview and work



Select Patient(s): Sven Handwerker

Select Value(s): Sleep quality

Select Period: 1/1/2023, 12:00:00 AM → 1/7/2023, 12:00:00 AM

These buttons are just for prototype preview:

Sleep quality ✓ Display Average Values

3	9	6	10.5	6	1.5	4.5
Sven H.						
3	9	6	10.5	6	1.5	4.5
Christian F.						
3	9	6	10.5	6	1.5	4.5

Steps ✓ Display Average Values

7500	10.500	6000	1500	6000	3000	9000
Sven H.						
7500	10.500	6000	1500	6000	3000	9000
Christian F.						
7500	10.500	6000	1500	6000	3000	9000

directly with the information without any prior technical knowledge. Furthermore, therapists have the option of viewing the measurement data in the same form as it is presented to patients in the patient app. This creates a shared perspective that facilitates communication between therapist and patient and enables better understanding of progress and challenges.

Visualization endpoints are designed to support consistent time-series rendering across heterogeneous data sources by applying unified normalization and timestamp alignment rules. This ensures that clinicians can compare signals and questionnaires on a common temporal axis.

3.4 Dependencies

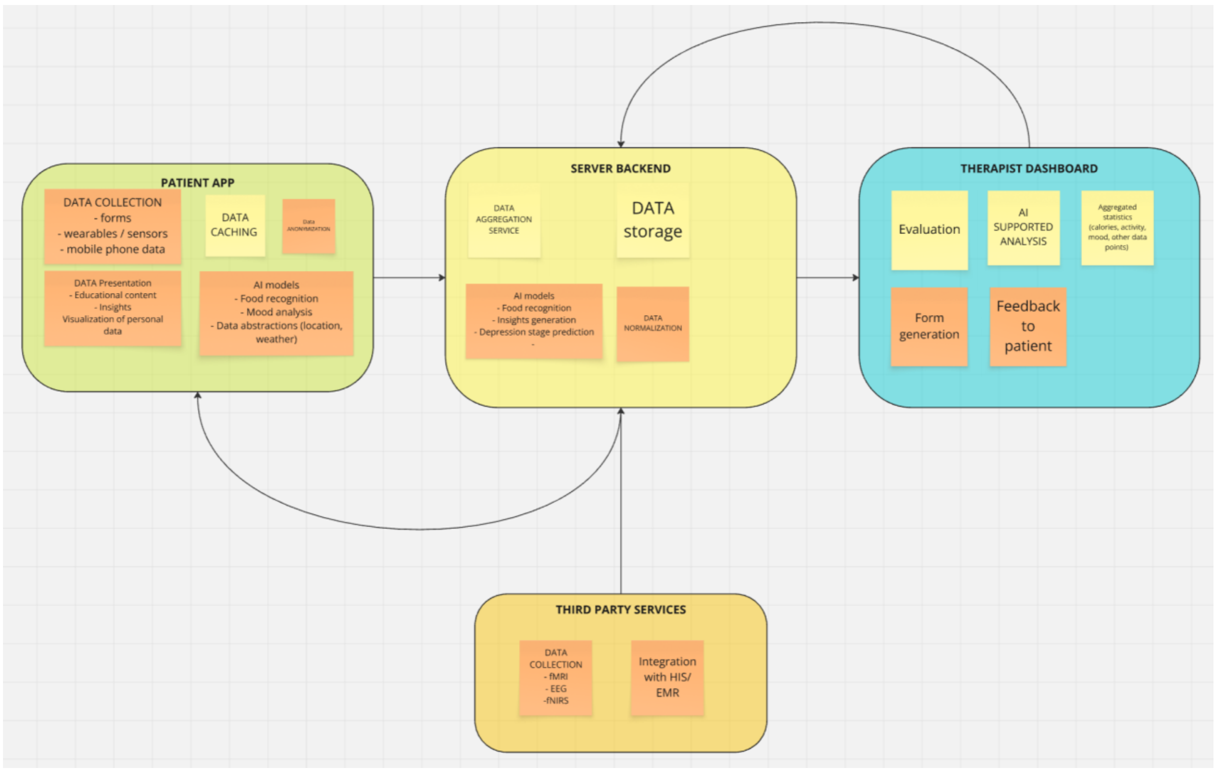
Angular and Node.js are used as the basis for the application, with the core evaluation logic located in the backend. This architecture also makes it possible to use elements from the patient app, which is built on a similar technological foundation. The visualization components of the project draw on external libraries such as the Bootstrap UI Kit and D3.js. These libraries provide robust tools and resources for creating interactive and visually appealing visualizations within the application. By using Bootstrap for UI design and D3.js for dynamic, data-driven visualizations, the project benefits from optimized development, a consistent technological basis, and an improved user experience for therapists and patients. In addition, Node-RED components can be integrated for curriculum creation, allowing processes to be flexibly modeled and automated. An editor is also available for content design, for example based on WordPress, which can be used to conveniently create, maintain, and customize teaching content.

3.5 Architecture

The therapist dashboard is designed purely as a user interface for therapists to view and enter information. It is architecturally clearly separated from the actual business logic and communicates exclusively with the backend via the API. All data processing, analysis, and AI modules are located in the server backend, so the dashboard itself does not contain any logic of its own, but merely functions as an interaction layer.

Access to the dashboard is secured by two-factor authentication, whose services are also part of the backend. This ensures that only authorized therapists are granted access. The process is designed so that the therapist first logs in via the dashboard, the authentication is checked by the backend, and then the data is provided via the API. The therapist's inputs are also forwarded to the backend via the API, processed there, and the results are transmitted back to the dashboard.

The architecture model thus follows a clear separation principle: frontend for usability and interaction, backend for processing, AI, and security. This structure ensures both high user-friendliness and the necessary security and scalability.


PATIENT APP
DATA COLLECTION - forms - wearables / sensors - mobile phone data
DATA CACHING
Data ANONYMIZATION
DATA Presentation - Educational content - Insights Visualization of personal data
AI models - Food recognition - Mood analysis - Data abstractions (location, weather)
SERVER BACKEND
DATA AGGREGATION SERVICE
DATA storage
AI models - Food recognition - Insights generation - Depression stage prediction -
DATA NORMALIZATION
THERAPIST DASHBOARD
Evaluation
AI SUPPORTED ANALYSIS
Aggregated statistics (calories, activity, mood, other data points)
Form generation
Feedback to patient
THIRD PARTY SERVICES
DATA COLLECTION - fMRI - EEG -fNIRS
Integration with HIS/ EMR

3.6 Deployment

Deployment as a microservice in a Docker container on a Kubernetes cluster

To deploy a service within a Docker container to a Kubernetes cluster, an image repository is essential. This repository will store the Docker images that Kubernetes will pull and run on its pods. Once the image repository is set up and linked to the Kubernetes cluster, the Kubernetes manifests is used to define the deployments, including the image to be used. The service is equipped with a Swagger interface. Swagger, also known as OpenAPI, provides a developer-friendly framework for describing the API of the service. This facilitates easier integration and control by providing a clear contract for the API and interactive documentation.

4 Therapy assistant – Patient App

The patient app serves as a centralized platform where patients can not only input and manage their health data but also access educational materials related to their conditions. This dual functionality supports both patient engagement and continuous learning, potentially leading to better health outcomes and more informed patients.

4.1 Type of data processed

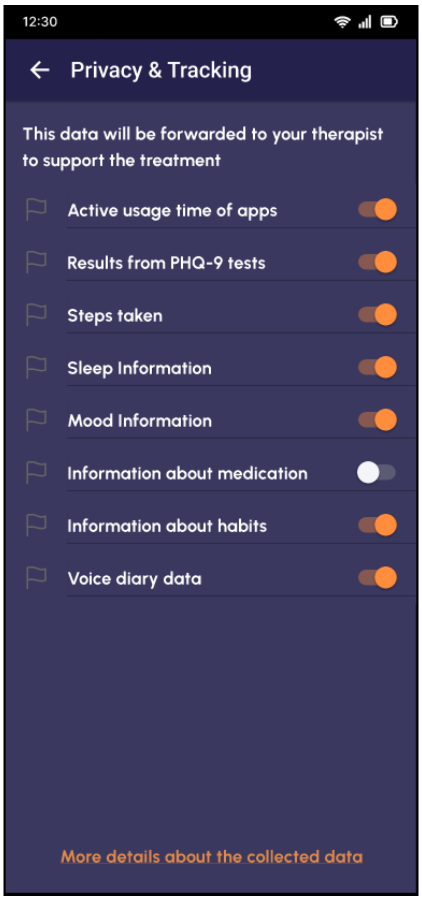
The system collects a wide range of patient data, including both medical and behavioral information. This includes technical metadata such as the last date of data transfer to the backend, as well as communication information about incoming and outgoing calls with start and end times and total duration. In addition, location data is stored, especially when a patient fills out a mood questionnaire, in order to later identify correlations between mood and location.

In addition to this contextual information, weather data is also taken into account, including general information on temperature, wind, humidity, and air pressure, as well as assumptions about the possible effects of the weather on mental health, such as headaches or concentration problems. The PHQ test also contributes to the patient's mental state, with responses stored on a scale and condensed into an overall score.

In addition, general master data such as name, age, gender, marital status, number of children, and level of education are available. This is supplemented by digital behavioral data, such as daily use of installed apps, sleep times, distances traveled, heart rate with minimum, maximum, and average values, and the number of steps taken per day. Information on medications and dietary supplements is also documented with name, quantity, and time period.

Lifestyle habits such as alcohol and cigarette consumption are recorded, as are the results of questionnaires on mood, energy, and stress levels, which can be supplemented with explanations. Finally, there is the option of keeping a personal diary in which patients can make written entries or voice recordings about their everyday lives, which can later be analyzed by therapists.

Given the sensitivity of patient-reported outcomes and behavioral signals, the app distinguishes between locally collected data and server-transmitted data, enabling privacy-driven control over what is shared. This supports ethical monitoring while preserving user autonomy.


12:30

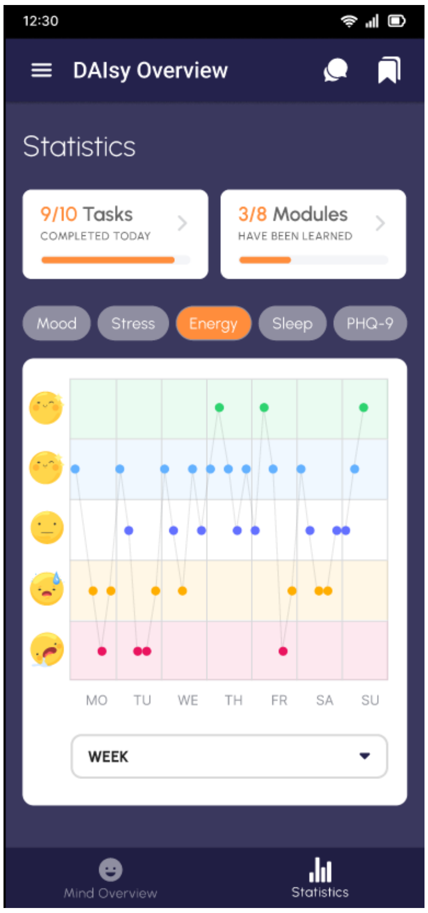
Privacy & Tracking

This data will be forwarded to your therapist to support the treatment

Active usage time of apps	on
Results from PHQ-9 tests	on
Steps taken	on
Sleep Information	on
Mood Information	on
Information about medication	off
Information about habits	on
Voice diary data	on

More details about the collected data

4.2 Opportunities for interaction

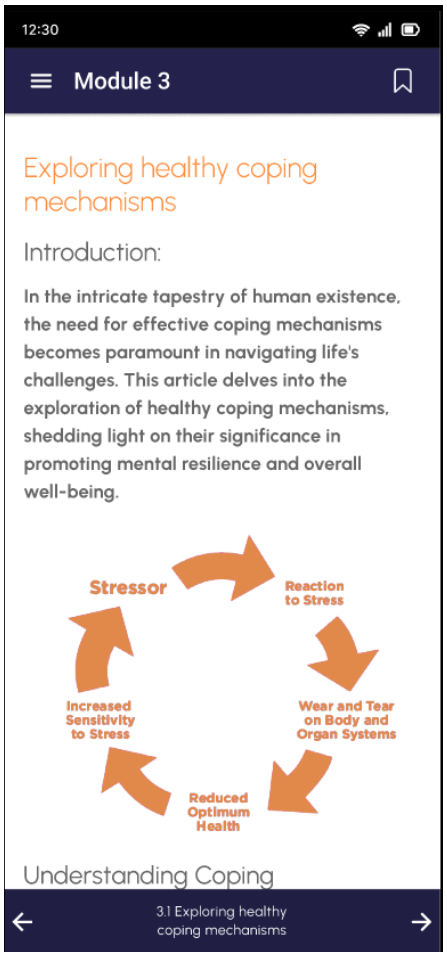


The patient's settings control both the collection and sharing of health information. A distinction is made between privacy settings and device permissions: In the privacy settings, the user actively decides whether sensitive data may be transmitted to the therapist's backend. Device permissions, on the other hand, affect the local level, such as whether the app is allowed to record data such as the use of other applications. This means that data may be collected locally but not forwarded to the server due to the privacy settings.

There is also a task management feature. Therapists can assign one-time tasks that are not repeated and can be edited, changed, or deleted by the patient. Patients themselves can define tasks with names, days of the week, and times, which are displayed on the device via notifications if required. These tasks can be modified or deleted at any time. Three fixed daily tasks for conducting mood checks in the morning, during the day, and in the evening are a special exception. These can be adjusted in terms of time, but cannot be deleted or changed in terms of the days of the week.

In addition, therapists can provide personalized learning content. This gives patients the opportunity to obtain additional information about their condition and how to manage it. This content is tailored to the individual patient and serves to support them in their everyday life, promote self-management, and deepen their understanding of their own health situation.

In addition, patients can view evaluations of their entries. This gives them the opportunity to track the development of their condition during therapy and better understand changes over time. This transparency supports both self-awareness and cooperation with the therapist. To improve engagement and data continuity, the app provides lightweight feedback views that summarize recent entries and trends, encouraging consistent self-reporting and device use across the monitoring period.



Exploring healthy coping mechanisms

Introduction:

In the intricate tapestry of human existence, the need for effective coping mechanisms becomes paramount in navigating life's challenges. This article delves into the exploration of healthy coping mechanisms, shedding light on their significance in promoting mental resilience and overall well-being.

Stressor
Reaction to Stress
Wear and Tear on Body and Organ Systems
Reduced Optimum Health
Increased Sensitivity to Stress

Understanding Coping

4.3 Interfaces

The patient app communicates with the backend system via standardized REST interfaces, which are secured by the Keycloak authentication system and ensure two-factor authentication. These interfaces form the central connection between the app and the server-side services, through which both the transmission of data marked as transmitted and access to the remote database take place.

The app uses various Capacitor plugins to connect to peripheral devices and internal data sources, enabling platform-independent development. This allows interfaces to devices and backend services to be used consistently, regardless of whether the app is running on iOS or Android.

The architecture clearly separates the local SQLite interface, where data is initially stored, from the REST interfaces to the backend, where the final transfer and processing takes place. This creates a modular structure in which interfaces secure and expand both internal data storage and external communication.

4.4 Dependencies

The architecture of the system is characterized by several technical dependencies, which arise primarily in the interaction between the front end, runtime environment, and interfaces. The front end is based on Angular and uses the Ionic framework to provide a cross-platform user interface. Ionic relies on WebView, which renders the Angular code and handles the logic.

In order for the app to access device functions such as sensors, cameras, or system information beyond pure web functions, there is a close dependency on Capacitor. This runtime environment connects the WebView with the native APIs of Android and iOS, enabling the use of native functions within the web code.

Capacitor plugins play a central role, acting as an interface between web code and native logic. They require both a native implementation in Java/Kotlin or Swift and a web interface for Angular. The stability and maintainability of the app therefore depends directly on the origin and quality of these plugins, whether they come from official sources, community contributions, third-party providers, or in-house developments.

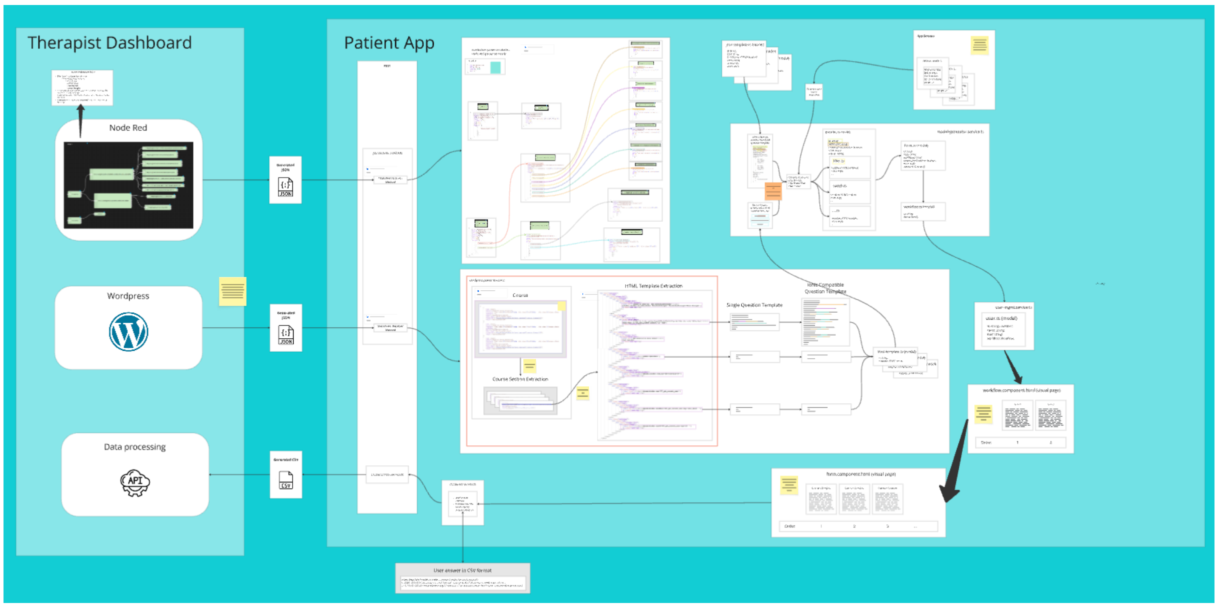
In addition, there is a dependency between the local SQLite database and the app logic. SQLite serves as a temporary storage for collected data, distinguishing between permanently relevant information and temporary data. This structure ensures that the app remains functional even without a direct connection to the outside world and that data can be managed consistently. To reduce the risk of gaps in monitoring, the synchronization logic supports deferred uploads and resilient transfer patterns, allowing data captured offline to be transmitted once connectivity is restored.

4.5 **Architecture**

The architecture of the patient app can be described as a multi-layered platform that clearly separates data collection, processing, and presentation. The front end features a mobile application developed with Angular and delivered across platforms via the Ionic Framework. Ionic ensures that the app behaves like a native application on Android and iOS by displaying UI elements in a platform-specific manner and integrating responsiveness rules for mobile devices. The app itself is based on a WebView that takes over the logic and display from the Angular code. Capacitor establishes the connection to the native device environment. Capacitor acts as a runtime environment that links the WebView to the native APIs of Android and iOS, enabling access to functions such as sensors, camera, or system information. A central component are the Capacitor plugins, which serve as a bridge between web code and native logic and provide both native implementation and a web interface. This allows developers to call functions in the Angular code that are forwarded internally to the native APIs. Depending on the origin of the plugins—official, community, third-party, or in-house development—the effort required for maintenance and compatibility varies.

Data processing within the app is supported by parser services that connect to external systems such as WordPress and Node-RED. Both systems provide data structures that are not directly usable and therefore need to be transformed by parser components in order to display them in a user-friendly way within the app. The parsers do not access the raw data of the systems, but obtain it from a central storage layer based on an SQLite database. This database stores and manages the structures and forms the basis for the work of the parser services. The Node-RED parser service extracts tab controls and individual nodes of the structure and makes them available as attributes. In addition, a BehaviourSubject is used to signal the completion of the parsing process. Since the parser does not work directly with the raw data but relies on the database, there is a close dependency on the storage service.

Some structures are currently not displayed correctly, so adjustments are necessary to remove superfluous or incorrect elements. The WordPress Parser Service processes two types of WordPress structures, namely contact forms and pages. After successful parsing, it provides the titles and contents of these elements and signals completion via its own attribute. Special features arise from the use of placeholders, which are replaced by real controls in the frontend. In addition, the service contains specific adjustments for DAIsy data that are not universally transferable. The parser ignores certain elements or uses them as anchor points in the parsing process. Here, too, there is a dependency on the SQLite database, as the data is not pulled directly from WordPress. In addition, the service is connected to the backend, which is responsible for loading e-learning data and other context operations.



The backend handles the actual business logic, data analysis, and AI modules. The data collected by the app and marked as “transmitted” is first stored locally in an SQLite database before being transferred to a MariaDB database in the backend. A distinction is made between permanently relevant data, such as privacy settings, and temporary data, which is deleted after transmission. The architecture thus follows a clear separation principle: the frontend is responsible for the user interface, input, and display; the capacitor layer connects web code with native functions; the parser services transform external data structures; the local SQLite database handles caching and management; and the backend takes care of storage, processing, and analysis. This structure enables modular further development, as the UI and interaction can be customized independently of the backend logic, while plugins can be flexibly extended or replaced and the parser services decouple external systems.

4.6 Deployment

Deploying an Ionic app on Android

Deploying an Ionic app on Android is a multi-step process that includes installing Android Studio, creating a release build of the app, signing the APK file, optimizing it, and finally publishing it to the Google Play Store. It is important that every step is done carefully to ensure that the app works correctly and complies with the guidelines of the Google Play Store. The use of commands such as 'ionic cordova build', 'keytool', 'jarsigner' and 'zipalign' are crucial. After optimizing the APK file and filling in all the necessary information in the Google Play Console account, the app can be made accessible to users worldwide.

5 MultiPy – Multimodal neurofeedback toolbox

5.1 Type of data processed

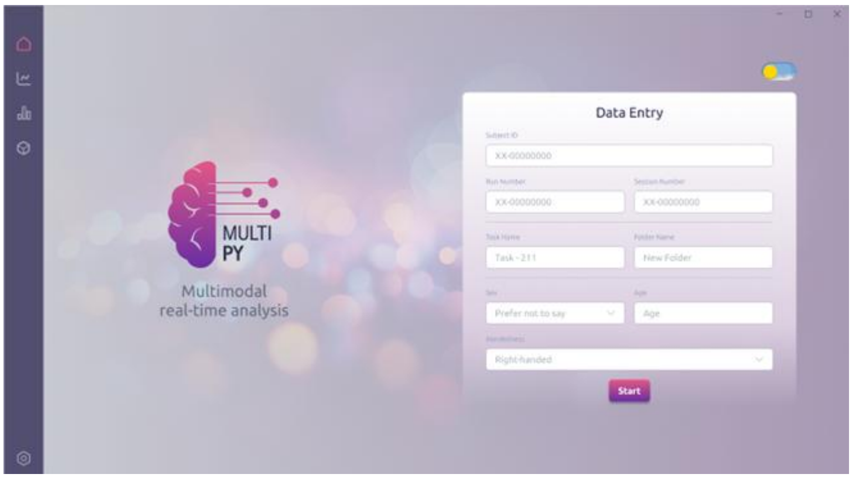
The neurofeedback component is designed for real-time processing of multimodal physiological data, with a focus on electroencephalography (EEG) and functional near-infrared spectroscopy (fNIRS). The EEG data consists of raw voltage signals across multiple channels, accompanied by event markers. The fNIRS data contains light intensity values that can be converted into optical density and hemodynamic concentration changes, such as oxygenated and deoxygenated hemoglobin (HbO and HbR). Both signal types can be enriched with metadata such as subject information, session identifiers, and time markers to ensure synchronization. This data is important for DAIsy because it provides the physiological basis for estimating cognitive or neural states, which can be used in adaptive feedback systems, therapeutic protocols, or experimental paradigms. The current phase focuses on reliably capturing, synchronizing, and preprocessing these data streams to support closed-loop interaction. Work is currently underway to extend these capabilities to more advanced analytics and cross-modal fusion.

5.2 **Opportunities for interaction**

The component provides interactive tools for exploring and manipulating the incoming data. Using the graphical user interface, users can, for instance, observe EEG and fNIRS signals in real time, select specific channels, and modify preprocessing parameters such as filter settings, artifact rejection criteria, and baseline intervals. The interface enables data aggregation across trials or time windows and supports transformations such as normalization, power spectrum calculation, and signal averaging. Users can create and test various feature extraction or modeling pipelines, adapting them to experimental requirements or participant responses. These interactive features are intended to enable an iterative process of exploration and optimization rather than fixed workflows and reflect the research-oriented and customizable nature of DAIsy. Additional interactive features, such as real-time threshold adjustment, and adaptive feedback calibration, are under development as part of ongoing work.

5.3 **Interfaces**

The component interacts with other parts of the system primarily through the Lab Streaming Layer (LSL) protocol, which manages both input and output data streams. EEG, fNIRS, and event marker signals are received via LSL inputs, while processed features and feedback signals are sent via LSL outputs. The system also supports interprocess communication via ZeroMQ for rapid data exchange and synchronization within the local environment. Data can be imported or exported in BIDS-compatible formats, ensuring compatibility with DAIsy's broader data management pipeline and with other neuroimaging tools. A Python API is implemented to enable external control and automation of preprocessing, feature extraction, and visualization components. The graphical user interface is based on PySide6 and QtPy and offers a flexible layout that can be embedded directly into the DAIsy main interface or run as a standalone window. Overall, the interface design aims to strike a balance between modularity and accessibility, enabling the integration of this component into various experimental or clinical setups.

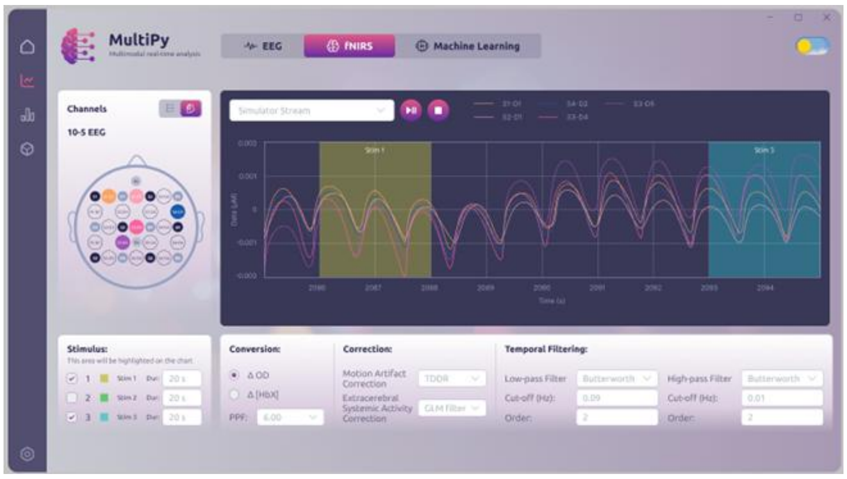


Data Entry

Subject ID	XX-00000000
Run Number	XX-00000000
Session Number	XX-00000000
Task Name	Task - 211
Folder Name	New Folder
Sex	Prefer not to say
Age	Age
Handedness	Right-handed

Visualization plays a central role in this component, providing continuous feedback on the state of the data and the performance of the processing pipeline. Users can monitor EEG and fNIRS signals in real time, examine spectral properties such as power spectral density, and visualize spatial distributions of brain activity using topographical or optode layout maps. The interface also presents diagnostic information, including data quality indicators. Current work is underway to integrate a multimodal visualization that combines EEG and fNIRS activity into a unified view, as well as to develop feedback dashboards that display key neurofeedback metrics and participant performance indicators. The visualization features are designed not only for real-time monitoring but also for interactive experiments. They allow users to examine specific time segments, adjust visualization parameters, and link visual representations directly to feedback control processes.

To facilitate monitoring in applied settings, the visualization layer can expose summary indicators (e.g., signal quality, artifact rates, and pipeline status) alongside real-time plots. These indicators support rapid assessment of whether observed changes reflect physiology or acquisition artifacts.



Stimulus:

✓	1	Stim 1	Dur:	20 s
	2	Stim 2	Dur:	20 s
✓	3	Stim 3	Dur:	20 s

Conversion: [✓ Δ OD] Δ [HbX]; PPF: 6.00

Correction: Motion Artifact Correction: TDDR; Extracerebral Systemic Activity Correction: GLM Filter

Temporal Filtering:

Low-pass Filter	Butterworth	High-pass Filter	Butterworth
Cut-off (Hz):	0.09	Cut-off (Hz):	0.01
Order:	2	Order:	2

5.4 Dependencies

MultiPy is based on a comprehensive set of Python libraries for data streaming, signal processing, visualization, and user interface development. Key dependencies include mne, mne-lsl, and pyxdf for EEG and fNIRS processing, pylsl for real-time data streaming, and numpy, scipy, pandas, and scikit-learn for computation, statistical analysis, and machine learning. Visualization and interface functions are supported by PySide6, QtPy, matplotlib, and pyqtgraph. Additional packages such as nilearn, nibabel, and mne-bids enable advanced fNIRS and neuroimaging operations, while joblib, tqdm, and psutil support performance monitoring and optimization. The dependency list is extensive and currently contains fixed versions to ensure cross-platform reproducibility and stability. As the component evolves, this list may be refined to improve performance and maintain compatibility with the broader DAIsy ecosystem.

5.5 Architecture

The component's architecture follows a modular and multi-layered structure. The acquisition layer manages data input via LSL streams and local file readers and synchronizes EEG, fNIRS, and event markers. The preprocessing layer applies filtering, artifact correction, and baseline normalization to both signal types. This is followed by a feature extraction and modeling layer, where derived metrics such as band power, connectivity indices, or hemodynamic responses are calculated in real time. The visualization layer dynamically presents this data via the graphical interface, while the feedback layer converts extracted features into control variables that can be fed back to DAIsy or external systems. The architecture also includes a plug-in and extension system that allows users to define their own processing or visualization modules without modifying the core code. The framework is currently being actively developed, with particular attention to reducing latency, improving cross-modality synchronization, and supporting hot-swappable pipeline components during live sessions.

5.6 Deployment

Deployment follows the standardized integration method for Python-based LSL components with GUI elements.

6 FitSprite Nutrition – Web Portal and Mobile App

6.1 Type of data processed

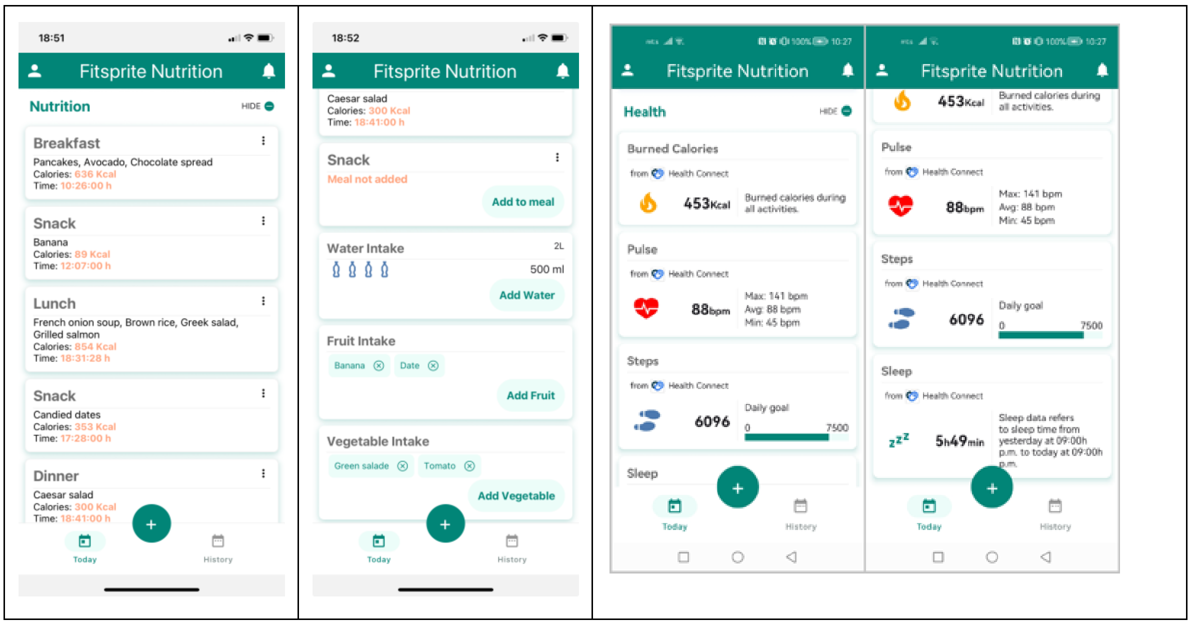
Nutrition data is collected by patient using various methods (e.g. AI-based food recognition, bar code scanning or manual input) and physical activity data is collected by smart wachtes. In the mobile app this data is processed by the insight generator, to generate insights for the user, whereas for the clinician view this data is aggregated (by day, per meal, etc.).

6.2 Opportunities for interaction

The clinician can filter/aggregate the data by meal type, over different time scales, as well as by whether binges occurred. Furthermore, the clinician has a possibility to send the patients notifications to notify them of behaviour. From a monitoring perspective, the portal supports clinician review by enabling aggregation across time windows and event tagging (e.g., binge occurrence), which can be used to relate behavioral patterns to intervention timing and patient-reported context.

6.3 Interfaces

Patients input their nutrition data via the mobile app. The physical activity data is collected on smart watches and is automatically included in the mobile app that sends data to the cloud to be presented on the web portal to the patient's clinician and stored for future reference and statistics calculation. The smart watches smart watches covered are: Samsung, Apple Watch, FitBit, Google Fit, Xiaomi, as well as any watches compatible with HealthConnect.



18:51
Fitsprite Nutrition
Nutrition HIDE

Breakfast
Pancakes, Avocado, Chocolate spread
Calories: 636 Kcal
Time: 10:26:00 h

Snack
Banana
Calories: 89 Kcal
Time: 12:07:00 h

Lunch
French onion soup, Brown rice, Greek salad, Grilled salmon
Calories: 854 Kcal
Time: 18:31:28 h

Snack
Candied dates
Calories: 353 Kcal
Time: 17:28:00 h

Dinner
Caesar salad
Calories: 300 Kcal
Time: 18:41:00 h

18:52
Fitsprite Nutrition
Caesar salad
Calories: 300 Kcal
Time: 18:41:00 h

Snack
Meal not added
Add to meal

Water Intake 2L
500 ml
Add Water

Fruit Intake
Banana Date
Add Fruit

Vegetable Intake
Green salade Tomato
Add Vegetable

Fitsprite Nutrition
Health HIDE

Burned Calories
from Health Connect
453Kcal Burned calories during all activities.

Pulse
from Health Connect
88bpm Max: 141 bpm Avg: 88 bpm Min: 45 bpm

Steps
from Health Connect
6096 Daily goal 0 7500

Sleep

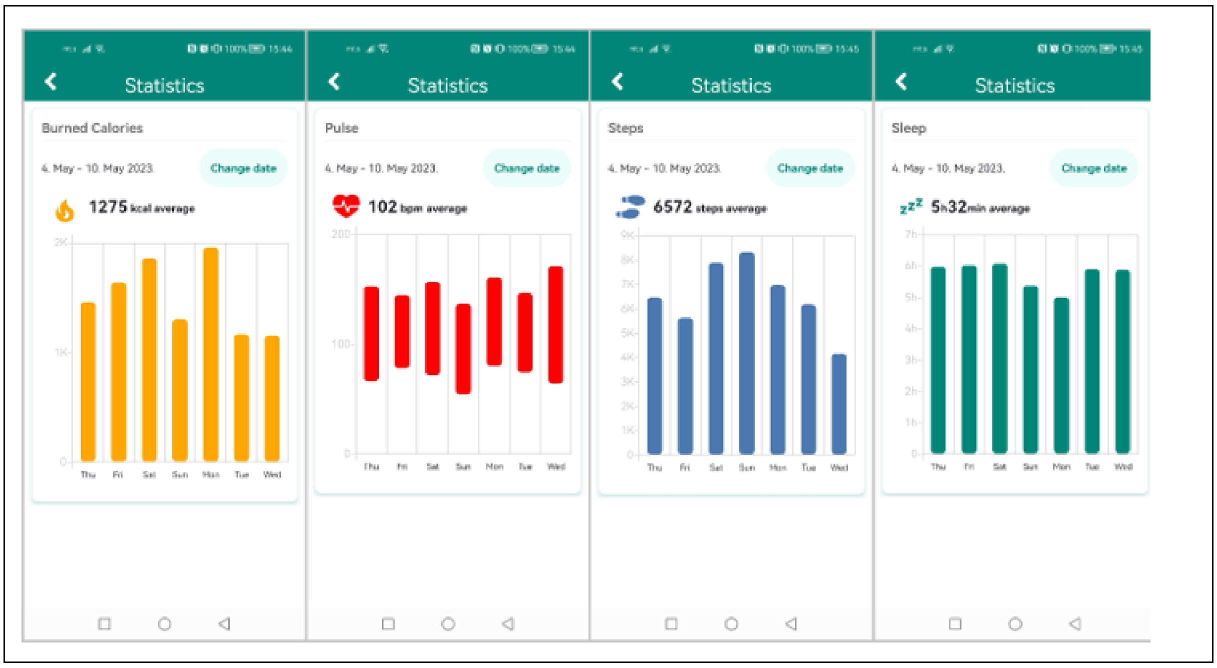
Fitsprite Nutrition
453Kcal Burned calories during all activities.

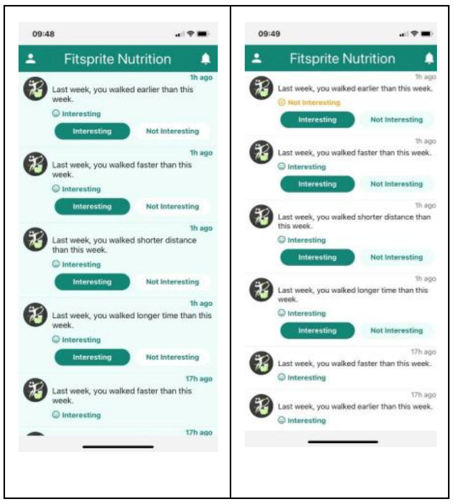
Pulse
from Health Connect
88bpm Max: 141 bpm Avg: 88 bpm Min: 45 bpm

Steps
from Health Connect
6096 Daily goal 0 7500

Sleep
from Health Connect
5h49min Sleep data refers to sleep time from yesterday at 09:00h p.m. to today at 09:00h p.m.

Patients can track the calories that they input as well as the calories that are burned during activities on the mobile app and insight generator offers users the statistics that the patient can mark as interesting. The insight generator learns constantly user preferences and adjusts its way of working accordingly. The web app user (clinician) can also use the platform to send notifications and recommendations to the patients.




09:48

Fitsprite Nutrition

1h ago
Last week, you walked earlier than this week.
Interesting
Interesting | Not Interesting

1h ago
Last week, you walked faster than this week.
Interesting
Interesting | Not Interesting

1h ago
Last week, you walked shorter distance than this week.
Interesting
Interesting | Not Interesting

1h ago
Last week, you walked longer time than this week.
Interesting
Interesting | Not Interesting

17h ago
Last week, you walked faster than this week.
Interesting

17h ago

09:49

Fitsprite Nutrition

1h ago
Last week, you walked earlier than this week.
Not Interesting
Interesting | Not Interesting

1h ago
Last week, you walked faster than this week.
Interesting
Interesting | Not Interesting

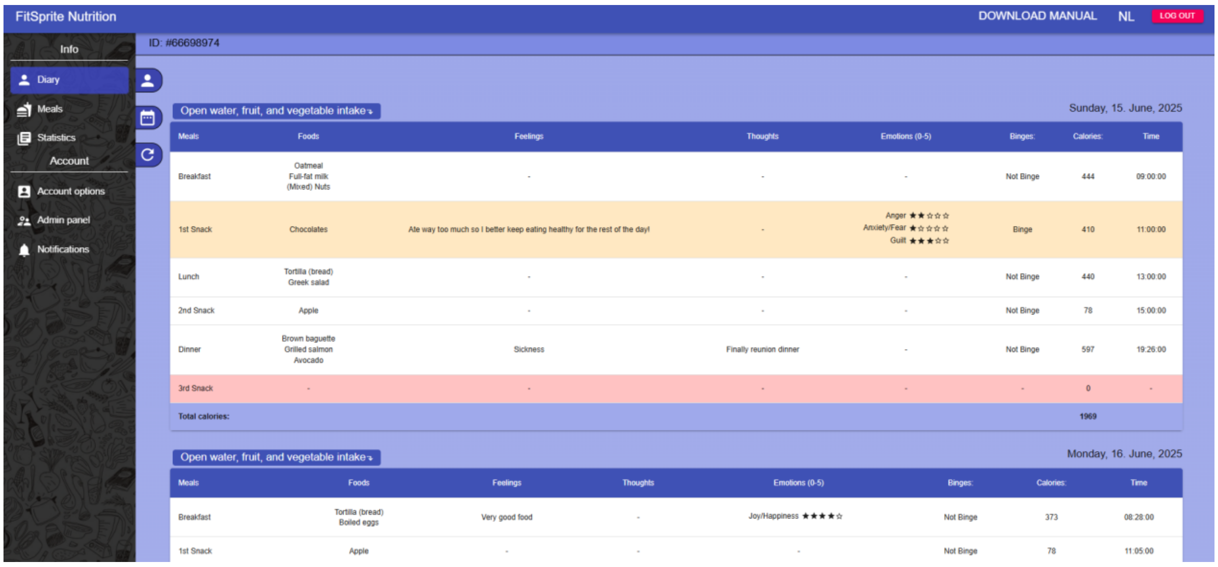
1h ago
Last week, you walked shorter distance than this week.
Interesting
Interesting | Not Interesting

1h ago
Last week, you walked longer time than this week.
Interesting
Interesting | Not Interesting

17h ago
Last week, you walked faster than this week.
Interesting

17h ago
Last week, you walked earlier than this week.
Interesting

The web portal presents an overview of the meals that a patient ate in a single day. For each meal it can be seen which food a patient had, how many calories the meal had, if there was a binge or not and the thoughts and feelings of the patient.



ID: #66698974

Open water, fruit, and vegetable intake

Sunday, 15. June, 2025

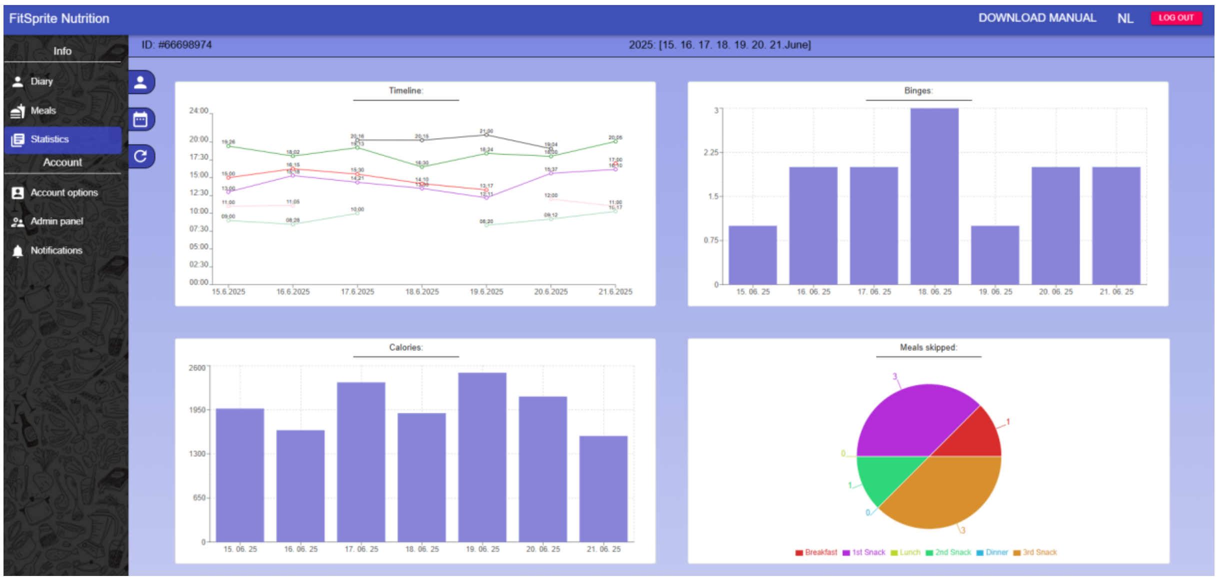
Meals	Foods	Feelings	Thoughts	Emotions (0-5)	Binges:	Calories:	Time
Breakfast	Oatmeal Full-fat milk (Mixed) Nuts	-	-	-	Not Binge	444	09:00:00
1st Snack	Chocolates	Ate way too much so I better keep eating healthy for the rest of the day!	-	Anger ★★☆☆☆ Anxiety/Fear ★☆☆☆☆ Guilt ★★★☆☆	Binge	410	11:00:00
Lunch	Tortilla (bread) Greek salad	-	-	-	Not Binge	440	13:00:00
2nd Snack	Apple	-	-	-	Not Binge	78	15:00:00
Dinner	Brown baguette Grilled salmon Avocado	Sickness	Finally reunion dinner	-	Not Binge	597	19:26:00
3rd Snack	-	-	-	-	-	0	-
Total calories:						1969	

Open water, fruit, and vegetable intake

Monday, 16. June, 2025

Meals	Foods	Feelings	Thoughts	Emotions (0-5)	Binges:	Calories:	Time
Breakfast	Tortilla (bread) Boiled eggs	Very good food	-	Joy/Happiness ★★★★☆	Not Binge	373	08:28:00
1st Snack	Apple	-	-	-	Not Binge	78	11:05:00

The statistics about the eating habits of the patient are also shown on the Web Portal. For example, at which time the patient usually has breakfast or lunch, if he had binges, the calorie intake per day and if he skipped some meals.



6.4 Dependencies

The system relies on a range of external libraries and frameworks that support machine learning, frontend and backend development, as well as user experience enhancements:

- **Frontend Development**: React, React Native, Material UI, React Router, Redux, Redux Thunk, Recharts, and Axios.
- **Utilities and Enhancements**: Babel, Lodash, date-fns, classnames, prop-types, and react-lazyload.
- **UI/UX Components**: Circular sliders, swipeable views, UI element libraries, vector icons, calendars, dropdowns, SVG rendering, video handling, and navigation frameworks for mobile and web.
- **Localization & Notifications**: Libraries for language localization and push notifications in the mobile application.

6.5 Architecture

Both mobile app and the web portal communicate with the backend via Web API's. Backend also contains AI models within a Docker Containers on the Hosting service. All patient's nutrition and physical activity data is stored in MySQL database.

6.6 Deployment

FitSprite Nutrition system is a stand-alone system. The patients are anonymized and not connected to the hospital Electronic Medical Record. Web portal is hosted at Digital Ocean hosting provider and Mobile apps are available at App Store and Google Play Store.

7 Major Depressive Disorder (Distinguishing Bipolar and Unipolar Depressive Disorder)

7.1 Type of data processed

This component processes data required for visualisation and monitoring of AI-supported analysis results for mental disorder subtype assessment.

The processed data includes:

- MRI imaging data accessed from PACS for authorised AI-analysis workflows
  .
- Prepared inputs generated during MRI data processing, which are used by the downstream feature extraction and further analysis stages.

- Region-based gray matter representations derived from structural MRI and organised at ROI level for analysis and visual interpretation.
- Clinical findings associated with the patient and relevant to diagnostic assessment, incorporated as contextual inputs during analysis.

- Analysis of outcomes, including diagnostic classification results and associated confidence information.
- Structured reporting outputs generated in the form of DICOM Structured Reports and stored back into PACS.

Raw imaging data remains under the control of PACS and is not permanently stored within the platform. Prepared inputs and derived representations are used exclusively to support AI-Inference and result presentation.

7.2 Opportunities for Interaction

The platform provides clear interaction points designed to support clinical workflows without introducing unnecessary complexity.

Key interaction opportunities include:

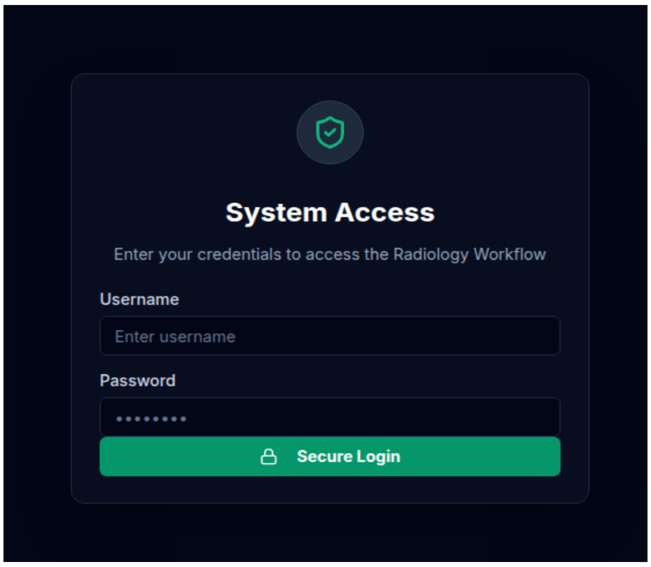
- Initiation of an AI-supported analysis through the Dashboard by an authorised clinician, using available clinical findings and imaging context.

- Passive monitoring of the analysis status while processing is executed as an orchestrated workflow.
- Automatic notification to the clinician when the analysis has completed.
- Direct access from the Dashboard to result review through the Viewer once processing is finished.
- Clinical inspection of imaging data together with derived ROI/tract overlays and analysis result summaries in the Viewer.
- Access to structured reporting outputs through standard clinical systems following completion of the workflow.

No continuous user interaction is required during processing. Clinical findings are incorporated as part of the workflow context without additional manual steps during execution.

7.3 Interfaces

The visualization and monitoring component interacts with other parts of the platform through well-defined interfaces. The Doctor Dashboard serves as the main entry point for initiating analysis requests, monitoring completion status, and accessing results once processing has finished. The Viewer supports detailed clinical review by presenting brain imaging together with derived segmentation overlays and analysis result summaries. Below are the sample pictures depicting the UI interfaces:



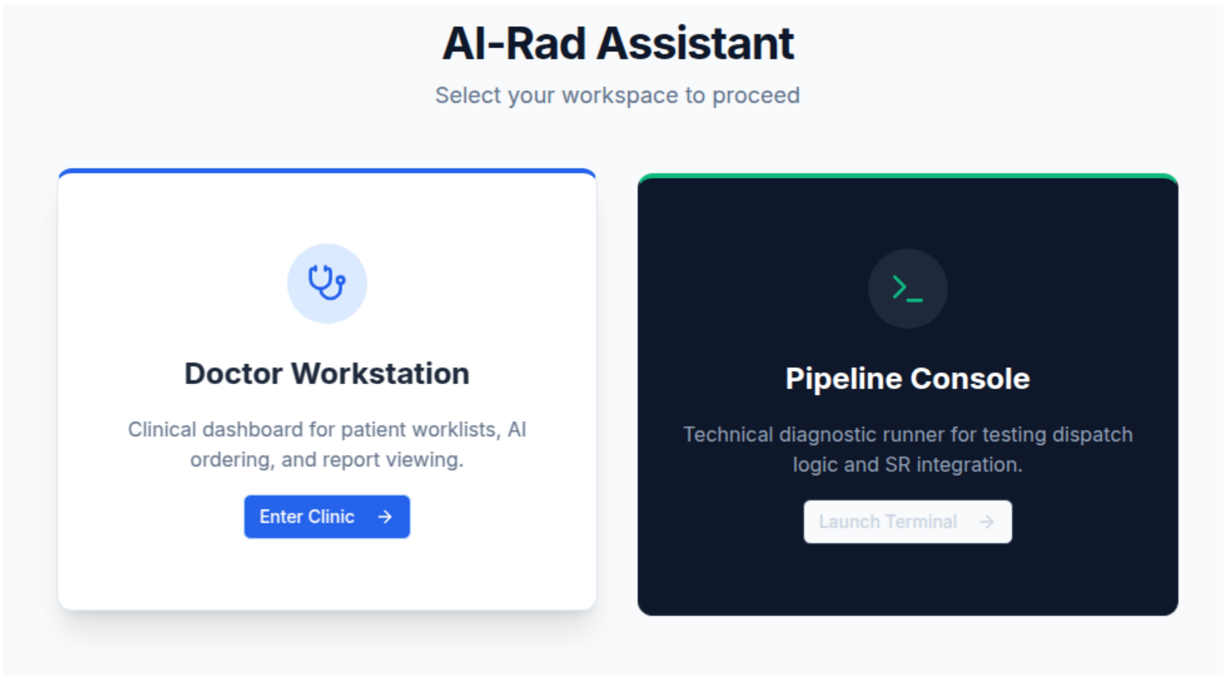
System Access

Enter your credentials to access the Radiology Workflow

Username: Enter username

Password: ••••••••

Figure 1. *Login screen of the Doctor Dashboard ensuring controlled access to analysis initiation and result review.*


AI-Rad Assistant

Select your workspace to proceed

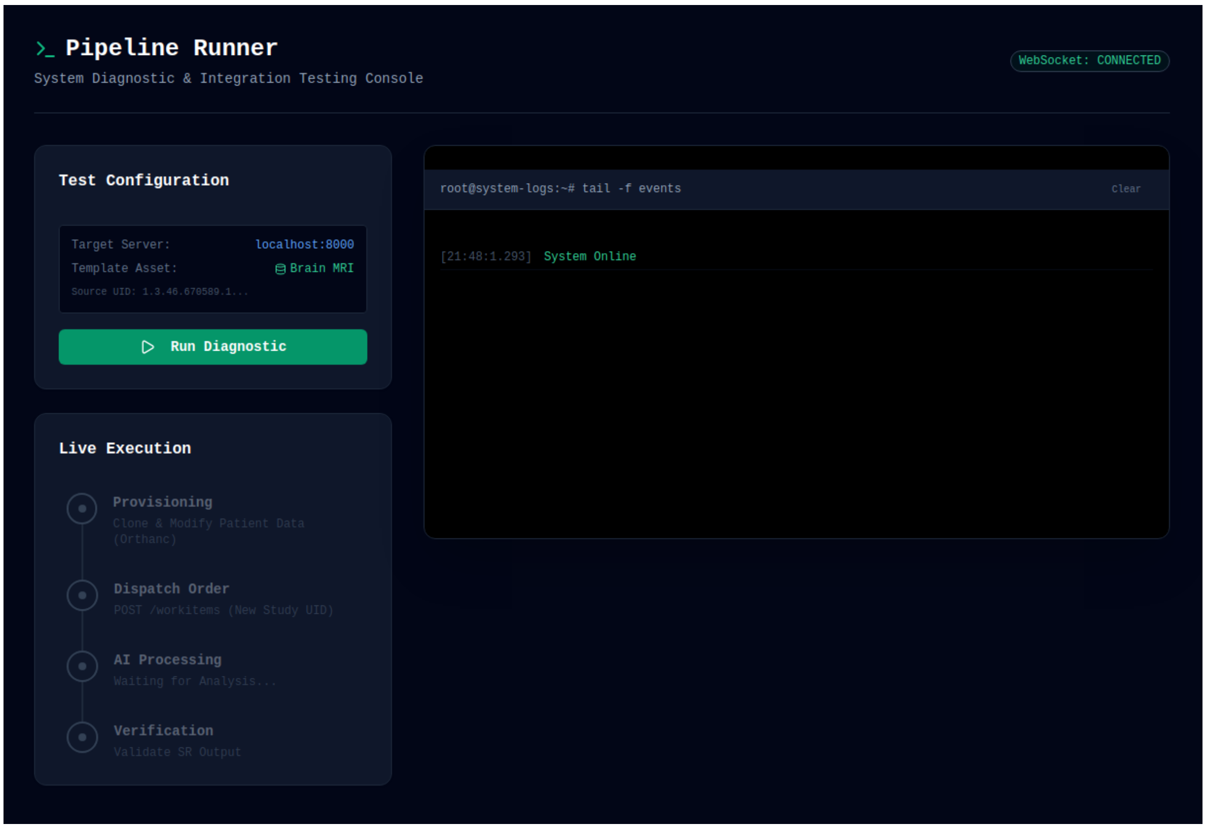
Doctor Workstation

Clinical dashboard for patient worklists, AI ordering, and report viewing.

Pipeline Console

Technical diagnostic runner for testing dispatch logic and SR integration.

Figure 2. *Workspace selection screen separating clinical access (Doctor Dashboard) from technical monitoring (Pipeline Console)*


>_ Pipeline Runner

System Diagnostic & Integration Testing Console

WebSocket: CONNECTED

Test Configuration

Target Server: localhost:8000
Template Asset: Brain MRI
Source UID: 1.3.46.670589.1...

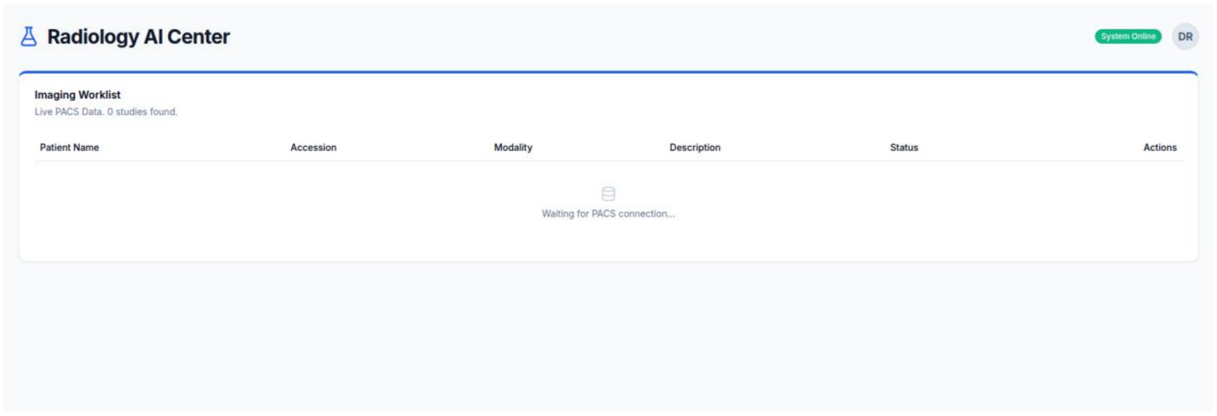
Live Execution

Provisioning — Clone & Modify Patient Data (Orthanc)
Dispatch Order — POST /workitems (New Study UID)
AI Processing — Waiting for Analysis...
Verification — Validate SR Output

```
root@system-logs:~# tail -f events                                    Clear

[21:48:1.293]  System Online
```

Figure 3. *Technical monitoring interface showing execution status and system events during pipeline operation.*


Radiology AI Center

System Online DR

Imaging Worklist

Live PACS Data. 0 studies found.

Patient Name	Accession	Modality	Description	Status	Actions

Waiting for PACS connection...

Figure 4. *Doctor Dashboard showing the imaging worklist and study status from PACS with patient context from HIS.*

All interfaces operate within authenticated and authorised workflows, consistent with the overall platform security design.

7.4 Dependencies

The visualisation and monitoring component depends on software and platform services that support clinical interaction and image-based review.

Key dependencies include:

- **VTK** (open-source) for volumetric brain rendering and overlay visualization in the Viewer.
- A **React-based** user interface stack for the Doctor Dashboard to initiate requests, display completion status, and provide access to result review.

7.5 Architecture

The visualization and monitoring component is organized around two user-facing elements: the Doctor Dashboard and the Viewer.

The Doctor Dashboard is used to initiate an analysis request and to present the workflow status to the clinician. When processing is completed, the dashboard notifies the user and provides a direct transition to result review.

The Viewer supports clinical inspection by presenting brain imaging together with derived segment overlays and a concise result summary. Visual review is performed without duplicating clinical imaging data or modifying original clinical records.

7.6 Deployment

The visualization and monitoring components are deployed as Dockerized services within the on-premise infrastructure of the clinical environment.

Deployment characteristics include:

- Independent deployment of dashboard and viewer components.
- Secure integration with backend processing services and clinical systems.
- Configuration aligned with local PACS and hospital information system endpoints.

This deployment approach allows visualization and monitoring capabilities to be integrated into existing clinical environments while preserving established workflows.