UNIVERSITY OF CALIFORNIA SAN DIEGO

Translating Physical Therapy Outcome Measures into an Open-Source Client-Side Web Application

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Computer Science

by

Daniel Ji

Committee in charge:

Professor Niema Moshiri, Chair
Professor Nadir Weibel
Professor Joel Okrent Wertheim

2026

The Thesis of Daniel Ji is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2026

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Niema Moshiri for his consistent support as the chair of my committee. Over the past four years, his guidance throughout my Bachelor's and now Master's has been invaluable. I am deeply grateful for his encouragement, advice, and belief in my potential. I would also like to thank him for his direction throughout the development of PhysioMeter and the writing of this thesis.

I would like to thank Karen George-Moshiri for her extensive collaboration on this project. As a licensed physical therapist, she provided clinical subject-matter expertise and reviewed the application throughout development. This work would not have been possible without her clinical expertise.

I would also like to thank Michael Puthoff and Sheryl Flynn of the APTA Geriatrics Annual Mobility Assessment Task Force, who authored the Annual Mobility Assessment, for their collaboration in ensuring the clinical correctness of PhysioMeter's implementation.

I would also like to thank my committee members, Professor Nadir Weibel and Professor Joel Okrent Wertheim, for their time and feedback.

Chapter 3, in part, is currently being prepared for submission for publication of the material. Ji, Daniel; George-Moshiri, Karen; Puthoff, Michael; Flynn, Sheryl; Moshiri, Niema. The thesis author was the primary investigator and author of this material.

Chapter 4, in part, is currently being prepared for submission for publication of the material. Ji, Daniel; George-Moshiri, Karen; Puthoff, Michael; Flynn, Sheryl; Moshiri, Niema. The thesis author was the primary investigator and author of this material.

Chapter 5, in part, is currently being prepared for submission for publication of the material. Ji, Daniel; George-Moshiri, Karen; Puthoff, Michael; Flynn, Sheryl; Moshiri, Niema. The thesis author was the primary investigator and author of this material.

Chapter 6, in part, is currently being prepared for submission for publication of the material. Ji, Daniel; George-Moshiri, Karen; Puthoff, Michael; Flynn, Sheryl; Moshiri, Niema. The thesis author was the primary investigator and author of this material.

VITA

2025 Bachelor of Science in Computer Science, University of California San Diego

2026 Master of Science in Computer Science, University of California San Diego

ABSTRACT OF THE THESIS

Translating Physical Therapy Outcome Measures into an Open-Source Client-Side Web Application

by

Daniel Ji

Master of Science in Computer Science

University of California San Diego, 2026

Professor Niema Moshiri, Chair

Physical therapists use standardized outcome measures to assess patient function and guide clinical decisions, yet adoption has been limited by fragmented workflows, scoring complexity, and limited automated interpretation. Existing digital tools are embedded in proprietary electronic health records, limited to specific measures, subscription-based, or run on remote servers.

This thesis presents PhysioMeter, a free, open-source, client-side web application for scoring and interpreting physical therapy outcome measures. Developed in collaboration with licensed physical therapists, PhysioMeter implements a declarative Measurements, Calculations,

and Interpretations pipeline in YAML configuration files, allowing measures to be added or modified without changes to the application code. All computation and storage occur within the clinician's browser; no patient data are transmitted to a remote server. It is also installable as a Progressive Web App that runs offline after first load.

The initial implementation includes an Annual Mobility Assessment protocol with nine outcome measures covering vital signs, gait speed, sit-to-stand performance, step tests, and the Timed Up and Go. Timing instruments, administration instructions, and real-time interpretation with age- and sex-stratified normative data are integrated into the workflow. Correctness was supported by 301 automated tests covering 23 clinically distinct patient scenarios and schema validation of all configuration files.

Because the architecture is not specific to any clinical specialty, PhysioMeter can be extended to neuro-rehabilitation, pediatric, sports, and orthopedic outcome measures through configuration changes alone.

Chapter 1

Introduction

1.1 Clinical motivation and problem statement

Physical therapists use standardized outcome measures to quantify patient function, track change over time, and guide clinical decisions. Performance-based measures—such as gait speed tests, the Timed Up and Go, and the 30-Second Sit to Stand—require the therapist to administer a physical task, record a numeric result, compute derived scores, and interpret the result against published normative data and clinical thresholds. In practice, this process is spread across paper forms, PDF reference documents, separate calculators and timers, and manual lookup of age- and sex-stratified normative tables.

This fragmented workflow is time-consuming, with the time required for scoring and interpretation identified as a primary barrier to outcome measure adoption [7]. It is also error-prone, as manual calculation and lookup introduce opportunities for arithmetic and transcription errors. A recent survey found that fewer than half of physical therapists in the United States believe that outcome measures are administered in a standardized way across the profession [8].

Prior work has shown that workflow integration can increase outcome measure utilization [9]. However, existing tools are either embedded in proprietary electronic health record platforms, limited to specific measure categories, subscription-based, or process patient health information on remote servers. We are not aware of an existing tool that combines client-side

execution, clinician-configurable measure definitions, automated interpretation with normative data comparison, and free open-source availability.

1.2 Proposed approach

This thesis presents PhysioMeter, a free, open-source, client-side web application for scoring and interpreting physical therapy outcome measures. PhysioMeter was developed in collaboration with licensed physical therapists and addresses the problems identified above through five design principles:

1. Client-side execution: All computation and data storage occur within the clinician's web browser. No patient health information is transmitted to or processed by a remote server. The same architecture makes the application installable as a Progressive Web App and usable offline after first load.

2. Declarative configuration: Outcome measures, scoring rules, and interpretation logic are defined in YAML configuration files rather than application code. Measures can be added and modified without changes to the application code.

3. Automated interpretation: The system applies clinical rules—including risk thresholds, normative data comparison, and mobility classification—automatically as data are entered, displaying severity-coded results with literature citations.

4. Workflow integration: Timing instruments (stopwatches, countdown timers), tallying controls, standardized administration instructions, and guided measurement navigation are built into the session interface, so the clinician does not need to switch between tools.

5. Correctness checking: A two-layer validation system (JSON Schema and cross-file reference checking) catches configuration errors at build time, and an automated test suite validates scoring and interpretation outputs against known clinical scenarios.

The initial implementation includes an Annual Mobility Assessment protocol with nine outcome measures. The architecture is not specialty-specific, so it can be extended to other physical therapy specialties and patient groups.

1.3 Evaluation summary

PhysioMeter's correctness is verified by 301 automated tests across patient fixtures and a systematic age/sex matrix, all configuration files pass schema and cross-reference validation, and 35 Playwright end-to-end tests exercise the full Annual Mobility Assessment session, the session-lock UX, and every SD-band zone and adverse-event flag in a real browser. The clinical content of the default preset was authored by the APTA Geriatrics AMA Task Force, and a licensed physical therapist reviewed the deployed application iteratively until outputs matched expected clinical interpretations. Adding a new outcome measure requires editing a few YAML files (plus an optional markdown instructions file) with no application code changes. Chapter 5 reports these results in detail.

The remainder of this thesis is organized as follows. Chapter 2 reviews outcome measures in physical therapy, barriers to adoption, and existing tools. Chapter 3 presents the system design, including the Measurements, Calculations, Interpretations pipeline and the privacy architecture. Chapter 4 describes the implementation, covering the five engine modules, session management, testing infrastructure, and contributor workflow. Chapter 5 presents the evaluation results. Chapter 6 discusses findings, limitations, and future work, and Chapter 7 summarizes contributions.

Chapter 2

Background and Related Work

This chapter reviews the role of standardized outcome measures in physical therapy, the barriers to their adoption, the landscape of existing digital tools, and the state of clinical decision support in rehabilitation. These topics motivate the design decisions described in Chapter 3.

A standardized outcome measure is a test with prescribed inputs and scoring rules that produces a numeric result. Using one in practice involves three steps: selecting the measure, administering it, and interpreting the result. Interpretation itself often has multiple layers: raw scores may need to be converted to derived values (e.g., walking time converted to gait speed in meters per second), derived values are compared against population norms stratified by age and sex, and clinical cutoffs are applied to classify risk (e.g., fall risk, frailty risk). Selecting which normative dataset to cite is itself part of interpretation: published sources for the same measure can disagree on means, standard deviations, and cut points, so the choice of reference data carries a clinical judgment of its own.

2.1 Barriers to adoption

Despite the established value of standardized outcome measures, adoption in clinical practice has been inconsistent. In a 2009 national survey, Jette et al. found that only 48% of US physical therapists reported using standardized outcome measures (16% in acute care), citing time constraints, scoring complexity, and interpretation difficulty as primary barriers [7]. Subsequent

work showed these barriers can be addressed through systematic intervention: McDonnell et al. combined EHR integration, consensus-based measure selection, educational sessions, and audit feedback in an acute care setting, raising usage from 16% to 100% [9]. More recent data [8] indicate that 97% of US physical therapists now use performance-based measures, but fewer than half (46–48%) agree that they are administered in a standardized way across the profession—suggesting that while adoption is widespread, consistency and rigor remain concerns.

Evidence on delivery format supports the case for digital workflows. A 2020 systematic review found that electronic administration of outcome measures is preferred by users, improves data quality, and shows completion times equivalent to paper [10], and a PT-specific validation study demonstrated that a mobile questionnaire app reproduced the psychometric properties of paper-administered instruments [12].

2.2 Digital scoring tools and clinical decision support

Several categories of tools exist for outcome measure scoring and interpretation in physical therapy and rehabilitation, each with distinct limitations.

2.2.1 Standalone scoring tools

Mobile Measures[1] is a mobile application for physical and occupational therapists that guides clinicians through test administration, calculates scores, and provides interpretation using published cutoffs and normative data. It supports over 40 patient populations and generates individualized reports. However, it is available only as a mobile app (not web-based), requires a subscription, and does not allow clinicians to configure or add new measures.

OrthoToolKit[2] provides free web-based calculators for orthopedic outcome measures including KOOS, HOOS, and QuickDASH. These tools compute scores from form inputs but

[1] https://mobilemeasures.org
[2] https://orthotoolkit.com

provide minimal interpretation beyond raw score calculation and are limited to orthopedic measures.

MDCalc[3] offers over 900 evidence-based clinical calculators across medicine. While broadly useful, it is physician-oriented rather than physical therapy-specific and does not provide PT-specific interpretation such as normative comparisons for rehabilitation outcome measures.

2.2.2 EHR-integrated outcome tracking

Several electronic health record (EHR) systems include built-in outcome measure tools. WebPT[4], a physical therapy EHR vendor, includes auto-scored outcome measurement tools and a tracking dashboard. Net Health TherapySource integrates with FOTO[5] (Focus on Therapeutic Outcomes) for predictive analytics and benchmarking against a database of over 44 million patient assessments. Jane App[6] and Proactive Chart include built-in outcome measure surveys with automatic scoring.

These tools share a common limitation: they are tightly coupled to their respective EHR platforms, require subscriptions, and process patient data on remote servers. They do not allow clinicians to configure measure definitions or add new measures. Their interpretation capabilities are generally limited to score tracking rather than clinical decision support with threshold-based classification.

2.2.3 Privacy and breach exposure in clinical tools

One further difference among existing tools is the privacy architecture. Clinical tools that process patient health information on remote servers carry the dominant healthcare privacy threat model: centralized-server breach. In 2023, 725 large breaches reported to the HHS Office for Civil Rights exposed over 133 million individuals' records, with hacking and other IT-related causes accounting for 79.7% of breach incidents and roughly 96% of records exposed [13, 5].

[3] https://www.mdcalc.com
[4] https://www.webpt.com
[5] https://www.fotoinc.com
[6] https://jane.app

By contrast, device loss and theft accounted for only 16 incidents across the same year (the lowest total recorded), a decline attributed to widespread encryption and cloud migration [5]. This gap matters because server-backed tools inherit the class of exposure that accounts for the overwhelming majority of healthcare records breached each year, whereas tools that keep patient data on the clinician's device shift the small remaining risk to the device itself.

2.2.4 Gaps in the existing landscape

A 2016 scoping review of 43 clinical decision support tools for patients with musculoskeletal disorders concluded that none had sufficient evidence to recommend for widespread clinical use [2], pointing to a gap between the available clinical knowledge (published thresholds, normative data, risk cutoffs) and the tools that apply it at the point of care. The tools and approaches described above leave several additional gaps that PhysioMeter is designed to address:

1. No standalone, client-side-only web tool: the scoring tools surveyed above are either mobile-only (Mobile Measures), embedded in EHR platforms (WebPT, Jane), or server-based analytics services (FOTO). None runs entirely in the browser with no server-side processing of patient health information.

2. No clinician-configurable measure definitions: the surveyed tools use proprietary, hard-coded measure libraries. None allows clinicians to define new outcome measures through structured configuration files.

3. Interpretation is rarely automated: most tools stop at score calculation. Few automatically contextualize scores with normative data, clinical cutoffs, and risk classifications in a unified view.

4. No workflow orchestration: clinical practice guidelines recommend sets of multiple measures administered together, but none of the reviewed tools allows clinicians to define and execute preset protocols with conditional dependencies between measures.

5. Cost barriers: FOTO is subscription-based with multi-thousand-dollar annual pricing tiers. Mobile Measures requires a subscription. EHR-integrated tools require platform lock-in. Free tools (OrthoToolKit) are limited in scope and interpretation.

Chapter 3

System Design

This chapter presents the design of PhysioMeter, a web application for scoring and interpreting physical therapy outcome measures. The design goals introduced in Section 1.2—client-side execution for privacy, declarative configuration of clinical content, correctness through validation, clinical workflow integration, and free open-source availability—shape the architecture, data model, and user interface described below.

3.1 Session model and data persistence

PhysioMeter organizes clinical data in a two-level hierarchy: patients and sessions. A patient record stores identifying information (name, date of birth, sex) and contains zero or more sessions. A session represents a single clinical encounter and stores a timestamp and all measurement data collected during the encounter. Data persists locally in the browser across page reloads and browser sessions, and the application auto-saves measurement data as the clinician enters values. The same hierarchy supports tracking across visits, and scoring rules that depend on patient demographics receive the patient's information automatically.

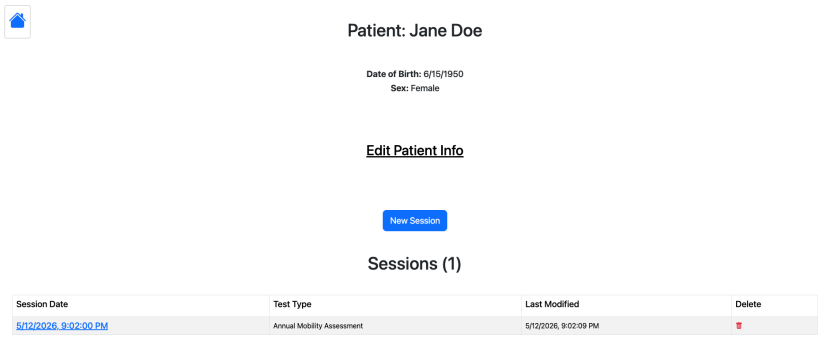

Figure 3.1. The patient page showing patient demographics and a completed session with timestamp and test type.

3.2 Conceptual model: Measurements, Calculations, Interpretations

PhysioMeter represents outcome measures through a three-stage pipeline: Measurements, Calculations, and Interpretations. Each stage is defined declaratively in YAML configuration files and processed by a corresponding engine at runtime.

3.2.1 Measurements

A measurement is a raw clinical data input collected during a patient encounter. Each measurement is defined by its input type, display label, validation rules, and optional administration instructions. PhysioMeter supports the following input types:

- Text: string, with optional pattern matching (e.g., blood pressure in "XX/YY" format).
- Decimal and Integer: numeric input with configurable bounds.
- Radio: single-selection from a list (e.g., sex, assistive device).
- Date: calendar date with validation (e.g., date of birth must be in the past).

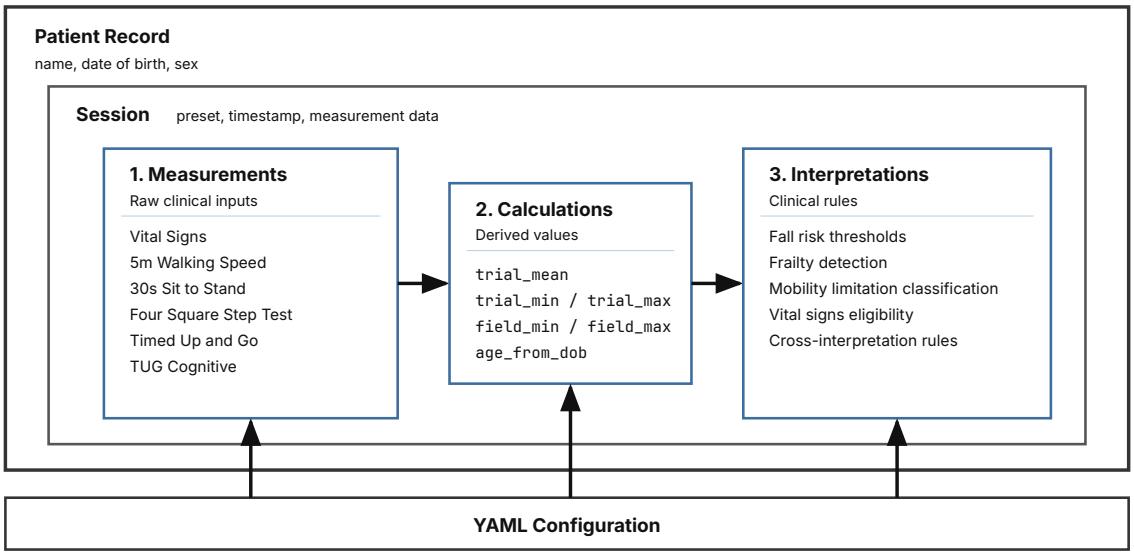

Figure 3.2. PhysioMeter system overview. A Patient Record contains Sessions, each of which uses a Preset to determine which outcome measures are available. Within a session, the three-stage scoring pipeline (Measurements, Calculations, Interpretations) processes clinical data. All pipeline stages are defined by YAML configuration files.

- Stopwatch: timed measurement with an integrated stopwatch control and configurable trial count.
- Countdown: timed measurement with an integrated countdown timer, used as a sub-field in 30-Second Sit to Stand.
- Fields: grouping of related inputs on one page (e.g., Vital Signs).

Measurements may also define disabled cases: conditions under which a measurement should not be administered. For example, all mobility tests in the Annual Mobility Assessment protocol are disabled when vital signs indicate the patient is ineligible for physical activity. Disabled cases are evaluated dynamically as data are entered, and clinicians may override them when clinical judgment calls for it.

3.2.2 Calculations

A calculation derives a single value from one or more measurement inputs. Calculations serve as an intermediate representation between raw inputs and clinical interpretations. The current implementation includes six calculation operations, though additional operations can be added to the calculation engine as needed:

- `trial_mean`: the arithmetic mean of all trial values across all instances of a measurement (e.g., the mean of two 5-meter usual walking speed trials).
- `trial_min`: the minimum trial value, used when the best performance is clinically relevant (e.g., the fastest time on the Timed Up and Go).
- `trial_max`: the maximum trial value across all instances.
- `field_min`: the minimum value of a specific named field within a grouped measurement (e.g., the best time from the Timed Up and Go Cognitive, which also records error counts).
- `field_max`: the maximum value of a specific named field within a grouped measurement.
- `age_from_date_of_birth`: computes the patient's current age in years from their date of birth.

Each calculation specifies the measurement it derives from, the operation to apply, and the number of decimal places for rounding. By defining calculations separately from both measurements and interpretations, the same measurement data can feed multiple calculations, and the same calculation can be referenced by multiple interpretation rules. A calculation can also be displayed alongside its source measurement as a computed value; for walking speed tests, for instance, the application converts the entered time in seconds to speed in meters per second and displays it in real time.

3.2.3 Interpretations

An interpretation applies a sequence of clinical rules to measurements and calculations, producing zero or more messages with associated severity levels and literature citations. Each interpretation is defined as an ordered list of rules; Table 3.1 summarizes the six rule types and their YAML keys.

A few concepts deserve mention beyond what the table captures. Message rules tag their output with one of four severity levels—concern, caution, normal, or info—which the UI renders as red, yellow, green, and gray respectively (Section 3.4). Threshold-lookup rules classify a value into one of three severity bands (green, yellow, or red) based on how far it falls from the age- and sex-matched population mean; each normative-data entry stores the population mean and standard deviation for its age/sex group, and the cutoffs for each band are derived by multiplying the standard deviation by configurable SD multipliers. Cross-interpretation rules let interpretations combine: the Annual Mobility Assessment interpretation, for example, calls the Vital Signs interpretation to decide whether to proceed with physical-activity testing.

Across all rule types, evaluation proceeds top-to-bottom against an accumulator-based context; the accumulator holds the variables bound by `let` rules and the messages emitted by `show_message` rules, which `otherwise` rules read to determine whether to fire.

A condition is a boolean predicate attached to a rule's `when` clause, evaluated against the patient's measurements, derived calculations, and any variables bound by earlier rules in the chain. A rule fires only when its condition is true. Conditions are evaluated by a general-purpose condition engine that supports numeric comparisons (`is_less_than`, `is_greater_than`, `is_between`), equality checks, pattern matching, existence checks (`is_missing`, `exists`), and logical combinators (`all_of`, `any_of`, `not`).

Table 3.1. Rule types supported by the interpretation engine.

Rule type	YAML keys	Effect
Skip	`when` + `then: skip`	Abort interpretation when inputs are missing
Message	`when` + `show_message`	Append severity-tagged message to the output
Variable	`let`	Bind a named value (e.g., unit conversion) for later rules
Threshold lookup	`lookup_threshold`	Classify against age/sex-stratified normative data
Otherwise	`otherwise`	Default message when no prior rule has emitted one
Cross-interpretation	`check_interpretation`	Branch on another interpretation's result

3.3 Presets and workflow orchestration

A preset defines a named subset of measurements that constitute a specific clinical protocol. When a clinician creates a new session, they select a preset, which determines which measurements are available and in what order they appear. Presets are defined in a YAML configuration file by specifying a key, display name, and an optional list of permitted measurement group keys. If no list is specified, all measurements are available.

PhysioMeter currently includes two presets: a Measurements preset that exposes the full configured measurement library without restriction, and the Annual Mobility Assessment preset, a curated nine-measure protocol (enumerated with input types and validation in Appendix A). Table 3.2 lists its measurements in administration order with their dependencies.

Both the safety gate and the conditional routing in the Annual Mobility Assessment preset are defined in YAML rather than application code. The vital-signs ineligibility rule (resting pulse above 100 bpm, oxygen saturation below 90%, systolic blood pressure above 180 or below 90 mmHg, or diastolic above 110 or below 60 mmHg) is defined once as a reusable `disabled_cases` entry and referenced by every downstream mobility test, so a single edit changes the gate for the whole protocol. The Four Square Step Test similarly defines

Table 3.2. Measurements in the Annual Mobility Assessment preset, in administration order.

Order	Measurement	Dependencies
1	Vital Signs	None (safety screen)
2	5-Meter Usual Walking Speed	Disabled if vitals ineligible
3	5-Meter Fast Walking Speed	Disabled if vitals ineligible
4	30-Second Sit to Stand	Disabled if vitals ineligible
5	Assistive Device Selection	None (determines FSST routing)
6	Four Square Step Test	Disabled if vitals ineligible or device is not None/Straight Cane
7	Modified Four Square Step Test	Disabled if vitals ineligible or device is None/Straight Cane
8	Timed Up and Go	Disabled if vitals ineligible
9	TUG Cognitive Dual Task	Disabled if vitals ineligible

an `action_button` that, if a patient cannot clear the apparatus, navigates the clinician to the Modified Four Square Step Test and bypasses its assistive-device restriction. Disabled cases are introduced in Section 3.2.1; clinicians may override either feature when clinical judgment calls for it.

3.4 UI/UX for clinical flow

The user interface follows the workflow of a clinician administering outcome measures during a patient encounter. Key design decisions include:

Linear measurement navigation.

Within a session, measurements are presented one at a time in a fixed order, with "Previous Measurement" and "Next Measurement" buttons for sequential navigation. A measurement selection page lists all available measurements for the current preset, allowing clinicians to navigate to any measurement directly. This shows one measurement at a time instead of all at once and matches the order in which the tests are administered.

Integrated clinical instruments.

Timing instruments are embedded directly in the measurement interface rather than provided as separate tools. Walking speed tests include a stopwatch with start, stop, and reset controls adjacent to the input field. The 30-Second Sit to Stand test includes a 30-second countdown timer. Integer measurements that track counts (e.g., number of sit-to-stands, TUG Cognitive error count) include increment and decrement buttons for quick tallying. Stopwatch and countdown values are auto-filled into the corresponding numeric field as the timer stops and remain editable, so a clinician can use the built-in timer for most administrations and override the recorded value if needed.

Standardized administration instructions.

Each measurement includes collapsible Markdown-formatted instructions specifying setup procedures, verbal scripts for the patient, and tester instructions. For the Annual Mobility Assessment preset these instructions follow the published administration materials [1] to promote standardized administration across clinicians and settings.

Real-time interpretation display.

Interpretations appear at the bottom of each measurement page and update dynamically as data are entered. Messages are color-coded by severity: red for concerns, yellow for cautions, green for normal results, and gray for informational messages. Each interpretation includes a clickable citation linking to its source material.

Standalone utility tools.

A utilities page provides five standalone tools—stopwatch, countdown timer, tally counter, calculator, and metronome—that are independent of the patient session system. These tools persist no data and are available for general clinical use. The metronome, for example, can be used for gait cadence assessment.

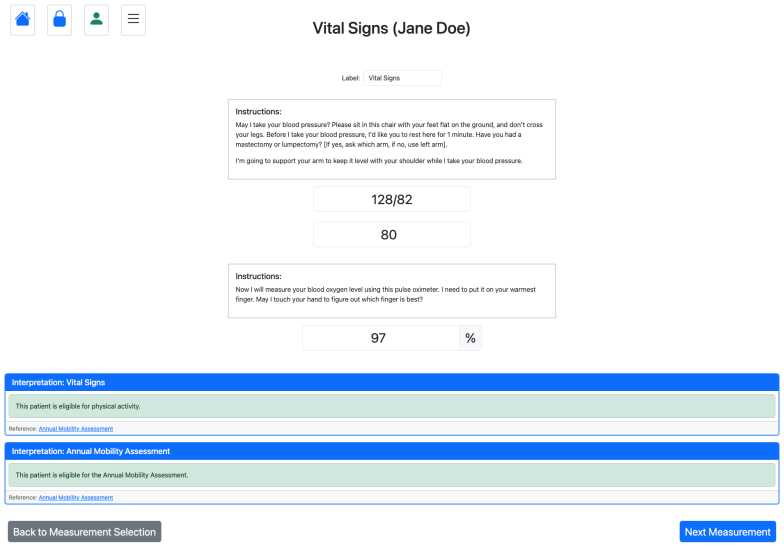

Figure 3.3. The Vital Signs measurement page with data entered. The interpretation at the bottom confirms the patient is eligible for physical activity based on pulse rate, blood pressure, and oxygen saturation values.

3.5 Summary and reporting

After completing a session, the clinician navigates to a summary page that aggregates all results into a single view. Results are grouped by measurement and include recorded values, calculated scores (e.g., gait speed derived from walking time), and severity-coded interpretation messages with citations to their clinical sources.

The summary page also serves as the source for data export. Clinicians can generate a PDF report or download session data as CSV files (one per section: measurements, calculations, and interpretations), enabling documentation and record-keeping outside the application.

3.6 Privacy and safety considerations

Because PhysioMeter runs entirely in the browser and is deployed as a static website, no patient data are transmitted over a network or stored on a remote server. This removes the application from the centralized-server breach exposure that dominates healthcare data incidents (Section 2.2.3).

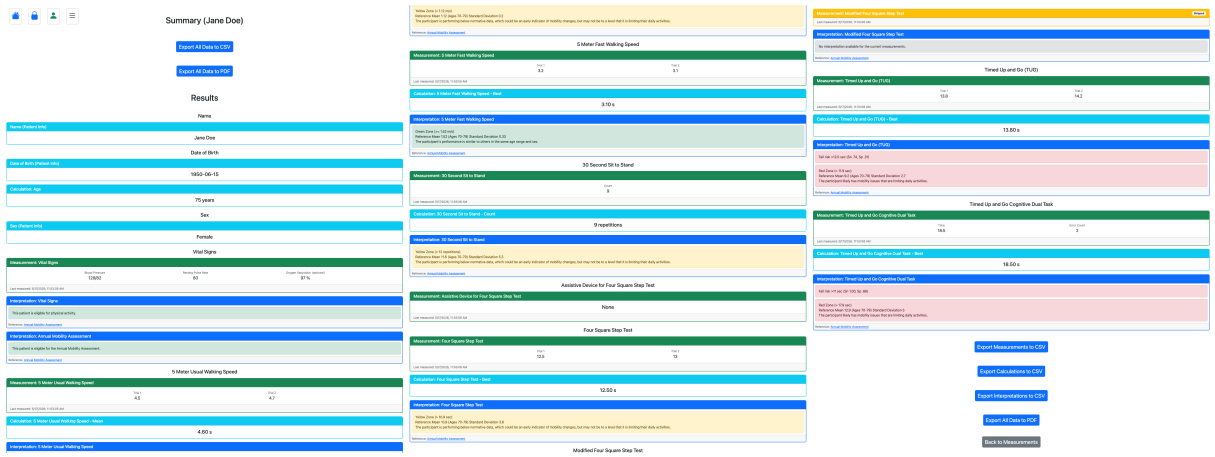

Figure 3.4. The summary page for a completed Annual Mobility Assessment session, shown as three vertical panels. Results are grouped by measurement and include patient demographics, recorded values, calculated scores, and severity-coded interpretation messages with citations.

Beyond not transmitting data to a server, the application sets no tracking cookies and makes no network requests after the initial page load. All patient data reside in the browser's IndexedDB on the clinician's device, under at-rest encryption tied to a session password (Section 3.6.1). Device-level risks (loss, theft, shared-device access) still depend on the operating environment, including disk encryption, screen lock, and device management.

3.6.1 Local encryption and session lock

Because all patient data lives on the device, the application encrypts each record before writing it to IndexedDB, using a key derived from a clinician-set session password. The goal is that a copy of the IndexedDB store taken without the password (an unauthorized viewer with same-browser access, browser-tools inspection without the password, or a backup of the browser profile from a lost or stolen device) does not yield readable patient information. The model does not protect against an attacker who holds the password, against malware running in the same browser context, or against an active attacker who can intercept keystrokes during unlock; those threats remain the responsibility of the operating environment.

Records are encrypted with AES-GCM-256 using a fresh 12-byte initialization vector per record. The encryption key is derived from the clinician's session password with PBKDF2-SHA256 at 600,000 iterations and a 16-byte per-installation salt, all through the browser's built-in `window.crypto.subtle` primitives without any third-party cryptography library. Whether an entered password is correct is checked by attempting to decrypt a known plaintext verifier written during initial setup, so the password itself is never stored. The derived key is held only in volatile memory for the lifetime of the unlocked tab, and is never written to IndexedDB, `localStorage`, `sessionStorage`, cookies, or window globals.

On first use, the application asks the clinician to choose a session password. The password is required to read or write patient records, but routes that do not touch patient data (the home page, the standalone timing instruments, the user guide, and the preset catalog) remain accessible while the session is locked. When the clinician navigates to a patient route while locked, the application redirects to a lock screen and returns to the requested route once the password is accepted. A manual lock control is available at all times, and after thirty minutes of inactivity the application clears the in-memory key automatically. Once locked, every database helper refuses to operate until the next unlock.

The session password is not recoverable, and if the clinician forgets it, the encrypted records on that device cannot be decrypted by anyone, including the application's maintainers. The only recovery option offered from the lock screen is a typed-confirmation reset that wipes the entire local database. This is an intentional tradeoff: the same property that prevents the application authors from reading patient data also prevents them from recovering it for a clinician who has lost the password.

3.7 Offline and installable deployment

PhysioMeter is packaged as a Progressive Web App (PWA). A web app manifest advertises installation metadata, and a service worker precaches the application bundle and static

assets on first load. After initial installation, the app can be launched from the home screen on mobile devices or from the applications menu on desktop, runs in a standalone window without browser chrome, and remains fully functional without a network connection. Because clinical logic, scoring, and interpretation all execute in the browser and all patient data live in IndexedDB, offline use imposes no functional restrictions: a clinician can administer a complete assessment, review interpretations, and export results on a tablet or phone with no connectivity. This is particularly useful for home visits, community screenings, and clinics with unreliable networks.

Chapter 3, in part, is currently being prepared for submission for publication of the material. Ji, Daniel; George-Moshiri, Karen; Puthoff, Michael; Flynn, Sheryl; Moshiri, Niema. The thesis author was the primary investigator and author of this material.

Chapter 4

Implementation

This chapter describes the implementation of PhysioMeter, covering the technology stack, the five engine modules that process YAML configurations into runtime behavior, the data persistence layer, and the testing and validation infrastructure.

4.1 Architecture and technology stack

PhysioMeter is a single-page application built with React and Vite. YAML configuration files are imported as JavaScript objects at build time, so all clinical logic defined in YAML is available to the application without runtime file parsing. The application is deployed as a static site to GitHub Pages via GitHub Actions, requiring no server infrastructure.

YAML configuration files are the central source of clinical logic. At build time, factory components load the YAML files and pass them to their corresponding engine modules, which transform the declarative configurations into runtime JavaScript objects (functions, validation rules, and data structures) consumed by React components.

The application has no runtime dependencies on external services. All network requests occur during the initial page load to fetch the application bundle; thereafter, the application operates entirely offline. A web app manifest and a Workbox-generated service worker (via the `vite-plugin-pwa` build plugin) make the app installable and precache the bundle so that

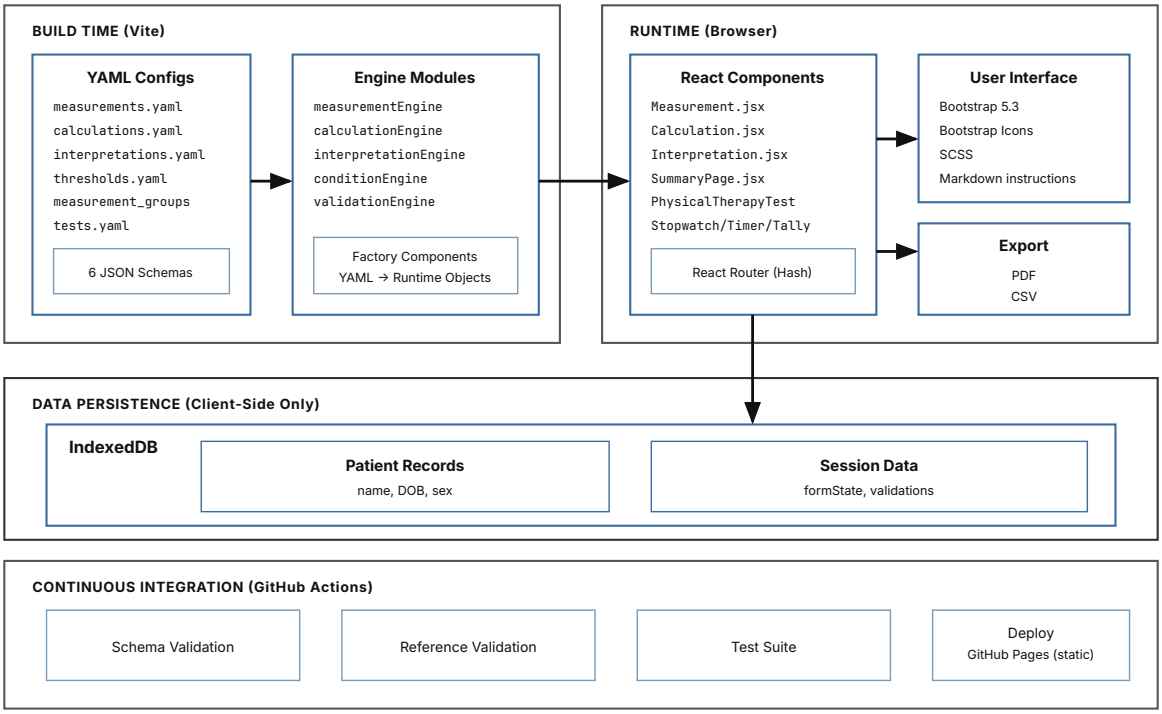

Figure 4.1. PhysioMeter architecture. YAML configurations are transformed into runtime objects by engine modules at build time. React components consume these objects to render the UI. Patient data are persisted in IndexedDB. Continuous integration validates configurations and runs the test suite on every push.

subsequent loads succeed without a network. Patient data are persisted in IndexedDB, and exports are generated entirely client-side.

4.2 Measurement library

The measurement engine (`measurementEngine.js`) transforms measurement definitions from `measurements.yaml` into runtime configuration objects. Each YAML entry is converted into a configuration that includes the input type, display label, placeholder text, validation function, optional administration instructions (which live in separate Markdown files referenced by name from the YAML), computed display functions, and disabled case evaluators.

Validation functions are built by the validation engine (`validationEngine.js`), which composes declarative validation rules into a single `(value)` $\Rightarrow$ `boolean` function. Supported validation rules include range checks (`is_between`, `is_between_exclusive`), integer checks, pattern matching with extracted sub-values, and inter-field comparison rules. For example, the blood pressure measurement validates the format "XX/YY" using a regular expression, extracts systolic and diastolic components, validates each is within the range 0–200, and enforces that systolic exceeds diastolic:

```
bloodPressure:
  type: text
  label: "Blood␣Pressure"
  placeholder: "blood␣pressure␣(XX/YY)"
  instructions: BloodPressure
  validation:
    matches_pattern: "^(\\d{1,3})/(\\d{1,3})$"
    extracted_values:
      systolic:
        group: 1
        type: integer
        is_between: [0, 200]
      diastolic:
        group: 2
        type: integer
        is_between: [0, 200]
    rules:
      - field: systolic
        is_greater_than_field: diastolic
```

Listing 4.1. Blood pressure measurement definition (excerpt from measurements.yaml).

4.3 Calculation engine

The calculation engine (`calculationEngine.js`) converts calculation definitions from `calculations.yaml` into objects containing a label, unit, and a `valueFunction(formState)` $\Rightarrow$ `value` closure. Each supported operation is implemented as a pure function:

- Aggregation: `trial_mean`, `trial_min`, and `trial_max` apply the corresponding aggregate function across all trials of a measurement.
- Field extraction: `field_min` and `field_max` extract values of a named field within a grouped measurement across all instances.
- Age computation: `age_from_date_of_birth` computes the patient's age in complete years, accounting for whether the birthday has occurred in the current calendar year.

4.4 Interpretation engine

The interpretation engine (`interpretationEngine.js`) is the most complex of the five engines. It executes interpretation rule chains by walking the rules array top-to-bottom, maintaining a local context with a mutable variables map and an accumulating messages array.

4.4.1 Condition engine

The condition engine (`conditionEngine.js`) is a general-purpose evaluator used by both the interpretation engine and the measurement engine's disabled case evaluators. It resolves values from three sources—measurements, calculations, and variables—and evaluates them against a set of comparison operators.

Logical combinators (`all_of`, `any_of`, `not`) allow conditions to be composed recursively. The `every_instance_of` operator evaluates a condition across all instances of a measurement, returning true only if every instance satisfies the condition.

4.4.2 Threshold lookup

The threshold lookup system implements age- and sex-stratified normative data comparison. Threshold tables in `thresholds.yaml` store the population mean and standard deviation per (sex, age-bracket) cell across the AMA age brackets; cutoffs for the three severity bands are derived at evaluation time by multiplying the standard deviation by configurable SD multipliers.

The lookup function resolves the patient's sex from a measurement and age from a calculation, finds the matching age bracket, and applies the classification cutoffs. The `compare_as` field determines the comparison direction: "higher_is_better" (e.g., walking speed) uses $\geq$ for the green zone and $\leq$ for the yellow and red zones, while "lower_is_better" (e.g., completion time) reverses the comparison direction. Classification messages include the cutoff value and reference statistics.

4.5 Presets

Preset configurations from `tests.yaml` are processed by the physical therapy test factory component, which generates React Router routes dynamically. For each preset, the factory filters the full measurement configuration list to include only those measurements whose group keys appear in the preset's `permitted_measurements` list. If no list is specified, all measurements are available. The filtered list is used to generate routes for each measurement page, the measurement selection home page, and the summary page.

4.6 Workflow utilities

The five standalone utility tools described in Section 3.4 (stopwatch, countdown timer, tally counter, four-function calculator, and metronome) are each implemented as an independent React component and persist no data. They are distinct from the session-integrated stopwatches, countdowns, and tally counters that are embedded within measurement pages and write their values directly into the form state.

4.7 Session management implementation

Patient and session data are stored in IndexedDB through a database abstraction layer (`DB.js`). The database uses two object stores: a `users` store keyed by patient UUID, in which each row is an encrypted-at-rest patient record, and a `meta` store holding the PBKDF2 salt and the encrypted password verifier used by the session-lock module (Section 3.6.1). Sessions are stored as a nested array within each patient record. The persisted row shape and the decrypted body are shown in Listing 4.2.

```
# users store row (persisted, one per patient)
uuid: string              # crypto.randomUUID()
iv: Uint8Array(12)        # per-record AES-GCM initialization vector
ciphertext: Uint8Array    # AES-GCM-256 encryption of the body below

# Decrypted body (in memory only)
uuid: string
name: string
dateOfBirth: string       # ISO date (YYYY-MM-DD), nullable
sex: string               # "Male" or "Female", nullable
createdAt: string         # ISO 8601
sessions:
  - uuid: string
    sessionTimestamp: string  # clinician-selectable
    createdAt: string
    testKey: string       # e.g., "annualMobilityAssessment"
    testName: string
    lastModified: string
    lastSaved: string     # null until explicit save
    data:
      formState: object
      validations: object
      disabledValues: object

# meta store row (singleton, id = "auth")
id: "auth"
salt: Uint8Array(16)            # PBKDF2 salt
verifierIv: Uint8Array(12)      # IV for the known-plaintext verifier
```

```
29  verifierCiphertext: Uint8Array  # AES-GCM encryption of "PHYSIOMETER_OK"
```

Listing 4.2. IndexedDB schema. The persisted `users` row stores only ciphertext; the decrypted body shown below it exists only in memory while the session is unlocked. The `meta` store is a singleton keyed by the literal id "auth".

Each read or write helper invokes a `requireKey` gate (Section 4.8) that throws if no session is unlocked. Auto-save is throttled to avoid excessive writes during rapid data entry.

When a session is loaded, the patient's demographic data (date of birth, sex) are automatically merged into the session's measurement data. This ensures that calculations requiring patient attributes (such as age computation from date of birth) and interpretations requiring demographic data (such as sex-stratified threshold lookups) receive the necessary inputs without redundant data entry.

4.8 Encryption and session lock implementation

The cryptographic primitives are isolated in `src/crypto.js`, which exposes pure functions for key derivation, encryption, decryption, random-bytes generation, and verifier round-tripping. The module imports nothing from the database or React layers; it operates only on the Web Crypto API.

The database module (`src/DB.js`) holds the derived key in volatile memory. Every read or write helper calls a `requireKey` gate that throws `LockedError` when the key is absent, making the locked state the database's default and ensuring that any code path that forgets to check the lock state still cannot read or write records.

A React context (`src/LockContext.jsx`) tracks lock state and exposes setup, unlock, lock, and wipe actions to the rest of the application. A `RequireUnlocked` wrapper guards patient, session, and measurement routes, redirecting to the lock screen when locked and returning to the requested page after unlock. The lock screen itself (`src/components/Lock.jsx`) handles first-time password setup, unlock, and the typed-confirmation reset; setup enforces a minimum

password length of eight characters. An inactivity hook (`src/hooks/useAutoLock.js`) resets a thirty-minute timer on user-input events and triggers a lock when it fires.

4.9 Testing infrastructure

The test suite is organized around the concept of parity testing: each test verifies that the output produced by the YAML-driven engines matches an independently derived expected value. This approach catches both engine bugs and YAML configuration errors, since any discrepancy between expected and actual output indicates a failure in one or both.

Configuration parity tests

(`configParity.test.js`) verify that the measurement group mapping, preset configurations, and their relationships are structurally correct. Tests confirm the correct number of entries, required fields, patient-level attribute marking, and that calculation and interpretation assignments match expectations.

Calculation parity tests

(`calculationParity.test.js`) test each calculation operation against known inputs and expected outputs. Coverage includes trial mean, trial min, trial max, field min, field max, and age computation. Tests verify behavior with empty form states, single trials, multiple trials, multi-instance measurements, and edge cases such as missing or non-numeric values.

Validation parity tests

(`validationParity.test.js`) test the validation functions built from YAML rules. Coverage includes range validation, integer checks, blood pressure pattern matching with extracted sub-values, inter-field rules (systolic $>$ diastolic), date of birth validation, and radio button option validation.

Interpretation parity tests

(`interpretationParity.test.js`) are the most extensive. They test the full interpretation rule chain execution against a suite of predefined patient fixtures representing diverse clinical scenarios:

- Normal cases: `HEALTHY_MALE_65`, `HEALTHY_FEMALE_75` — patients with values in the normal range.
- Risk cases: `AT_RISK_FEMALE_92`, `VERY_SLOW_MALE_75` — patients who trigger fall risk and frailty interpretations.
- Ineligibility cases: six fixtures testing each vital signs ineligibility condition individually (high pulse, low SpO_2, high/low systolic, high/low diastolic).
- Edge cases: empty form state, null measurements, vitals-only data, patients without demographics, patients below the normative table age range (`YOUNG_PATIENT`), and multi-instance measurements.
- Age/sex matrix: a systematic matrix of 10 age bracket $\times$ sex combinations (`AGE_SEX_MATRIX`), each tested against all threshold tables to verify correct zone classification across the entire normative data range.

Encryption unit tests.

A separate unit-test file exercises the encryption helpers in `crypto.js` (Section 4.8): round-trip encryption, verifier round-trips, and wrong-password rejection.

End-to-end testing.

Three Playwright specs run in a Chromium browser. `e2e/user-guide-walkthrough.spec.js` drives a full Annual Mobility Assessment session: creating a patient, filling each of the nine measurements, and reaching the summary page. `e2e/lock-flow.spec.js` exercises the session-lock UX (setup, unlock, manual lock, and reset). `e2e/interpretation-coverage.`

`spec.js` drives every interpretable measurement through inputs sized to land in each SD-band zone and each documented adverse-event flag, verifying the rendered classification text across age and sex brackets. The parity tests operate on form-state objects; the Playwright suite covers routing, persistence, and the rendered UI.

4.10 Schema validation and continuous integration

Configuration correctness is enforced at two levels through an automated validation pipeline invoked by `npm run validate:config` and run in continuous integration (GitHub Actions) on every push and pull request.

JSON Schema validation.

Six JSON Schema files (one per YAML configuration file) define the structural requirements for each configuration type. Schema validation is performed by `ajv-cli` using the Draft 2020-12 specification. This layer catches type errors, missing required fields, invalid enum values, and structural violations.

Cross-file reference validation.

A custom validation script performs 13 referential integrity checks across the six YAML configuration files. These checks verify that every measurement, calculation, and interpretation is correctly registered in a group; that cross-file references (e.g., a calculation referencing a measurement, an interpretation referencing a threshold table) resolve to valid targets; that keys are unique; and that instruction file references resolve to existing files on disk.

4.11 Contributor workflow for adding outcome measures

Adding a new outcome measure to PhysioMeter requires editing YAML configuration files and optionally adding a Markdown instruction file. No JavaScript or JSX code changes are needed. The workflow involves four steps:

1. Define the measure: add the measurement definition in `measurements.yaml` (input type, label, validation rules), its calculations in `calculations.yaml`, and its interpretation rule chain in `interpretations.yaml`. If normative data are available, add age/sex-stratified threshold data in `thresholds.yaml`. Optionally write a Markdown file with standardized administration instructions.

2. Register and assign: create an entry in `measurement_groups.yaml` mapping the measurement to its calculations and interpretations, and optionally add it to a preset in `tests.yaml`.

3. Validate: run the validation pipeline to check both schema conformance and cross-file referential integrity.

4. Test: verify that existing tests still pass and add new test fixtures for the new measure.

This workflow is accessible to contributors with clinical expertise but limited programming experience. The YAML syntax requires no knowledge of JavaScript, React, or the application's internal architecture, and a step-by-step tutorial for editing these configurations is provided in the repository. Schema validation provides immediate feedback on structural errors, and cross-file reference validation catches broken links between configuration files before the application is built.

Chapter 4, in part, is currently being prepared for submission for publication of the material. Ji, Daniel; George-Moshiri, Karen; Puthoff, Michael; Flynn, Sheryl; Moshiri, Niema. The thesis author was the primary investigator and author of this material.

Chapter 5

Evaluation

This chapter reports the results of the automated correctness tests and the iterative clinical review by licensed physical therapists.

5.1 Correctness of scoring and interpretation

To evaluate the accuracy of the Annual Mobility Assessment implementation, an automated test suite (Section 4.9) was developed in which the expected values reflect the clinical specifications supplied by the PT collaborators. The suite acts as a regression guard against drift between the YAML-driven engines and those specifications.

5.1.1 Test coverage

The 301-test suite covers the scoring and interpretation pipeline, from input validation through calculation and rule execution. Table 5.1 summarizes the coverage by engine component.

All 301 tests pass across the fixtures described in Section 4.9. All six YAML configuration files pass JSON Schema validation, and all 13 cross-file reference checks pass, confirming the structural and referential integrity of the configuration.

Table 5.1. Test suite coverage by engine component.

Component	Tests	Coverage description
Configuration parity	17	Group mappings, preset structure, calculation/interpretation assignments
Calculation parity	47	All 6 operations across single/multi-trial, multi-instance, and edge cases
Validation parity	154	Range, integer, pattern, extracted sub-values, inter-field rules, date
Interpretation parity	76	All interpretation rule chains across patient fixtures and an age/sex matrix

5.2 Clinical review of interpretations and workflow

The clinical content of the Annual Mobility Assessment preset (the selection of outcome measures, the interpretation rules and age- and sex-stratified threshold data, the standardized administration instructions, and the workflow structure including measurement order, vital-signs gating of mobility tests, and FSST/Modified-FSST routing) was authored by the APTA Geriatrics Annual Mobility Assessment Task Force [1] and adopted into PhysioMeter through configuration. A licensed physical therapist with geriatric clinical expertise reviewed the deployed application against representative patient scenarios and provided feedback on interpretation accuracy and workflow design, working iteratively with the thesis author whenever a discrepancy appeared between the application's output and the expected clinical result. The relevant YAML configuration was corrected and the process continued until the outputs matched the expected clinical interpretations for all tested scenarios.

Chapter 5, in part, is currently being prepared for submission for publication of the material. Ji, Daniel; George-Moshiri, Karen; Puthoff, Michael; Flynn, Sheryl; Moshiri, Niema. The thesis author was the primary investigator and author of this material.

Chapter 6

Discussion

6.1 Interpretation of findings

PhysioMeter implements outcome-measure scoring and interpretation as a declarative YAML pipeline running entirely in the browser (Chapter 3). The choice of YAML matters because clinicians need to review the rules; running entirely in the browser matters for privacy.

YAML was chosen over alternatives (JSON, TOML, a custom DSL, and formal clinical-rule languages such as Arden Syntax [11] and HL7 Clinical Quality Language [4]) because a clinical collaborator without programming experience had to be able to read and edit the configuration files directly. YAML supports comments, threshold-table diffs are line-by-line, and a clinician revisiting the rule chain a year later can trace a measurement through its calculation, threshold lookup, and emitted message without reading application code. The cost is YAML's lack of strict structure: the validation pipeline (Section 4.10) is doing work that a more constrained format would do for free. A stricter format (a custom DSL, a heavily typed JSON schema, or a clinical-rule language designed for portability between vendors) would have required programming knowledge or specialized tooling to read and edit, defeating the goal of letting clinicians audit the rules themselves.

Running entirely in the browser keeps patient data off any remote server, sidestepping the centralized-server breach exposure that dominates healthcare data incidents (Section 2.2.3). The installable PWA bundle (Section 3.7) extends the same property to settings without reliable

network connectivity, with neither a network dependency nor the administrative overhead of a managed server.

The architecture does not, by itself, make a deployment HIPAA-compliant. HIPAA compliance is a property of the covered entity's overall program rather than of any individual application, and depends on administrative, physical, and technical safeguards together: device management, workforce training, BYOD policy, export handling, and access controls. Clinics deploying PhysioMeter should confirm with their Privacy and Security Officers that the surrounding program is HIPAA-compliant. The records the application stores on the device are encrypted at rest under a key derived from a clinician-set session password (Section 3.6.1); the administrative and physical safeguards around it are the adopter's.

Alongside that architecture, the tool is built to fit the clinical workflow. Embedding timing instruments, administration instructions, and real-time interpretation in one session addresses the time and scoring-complexity barriers identified in the adoption literature [7, 9]; whether this translates into measurable adoption gains would require a dedicated field study.

The conceptual model itself is not specialty-specific: any outcome measure expressible as inputs, calculations, and rule-based interpretations can be added through configuration, provided the necessary scoring rules and normative data exist in the literature.

6.2 Limitations

Limited measure library.

The current implementation includes one clinical preset (Annual Mobility Assessment) with nine outcome measures focused on mobility and fall risk in older adults. The immediate clinical value of the tool is bounded by that scope and grows as additional measures and presets are contributed.

Interpretation scope.

The interpretation engine classifies individual sessions against published thresholds but does not compare results across a patient's prior sessions to flag clinically meaningful change. The data model supports stored session histories, so this is an interpretation-engine limitation rather than a data one.

Client-side data persistence limitations.

IndexedDB storage is subject to browser-imposed storage limits and is tied to a specific browser on a specific device. Patient data cannot be synchronized across devices or shared between clinicians without manual export and re-import. Browser data clearing or device loss results in data loss. These constraints follow from the client-side architecture and would need to be addressed for clinical settings that require multi-device access or centralized records. A forgotten session password is another failure mode along the same axis (Section 3.6.1): the only recovery option is the typed-confirmation reset, which discards the data.

EMR integration.

Results live in the local database and in exported PDF and CSV files. A clinic that uses an electronic medical record still has to bring those results into the record by hand, since PhysioMeter does not write to any EMR directly. An export format the local EMR can ingest, or a deliberate integration path, would remove the manual re-entry step.

Application updates require a network.

A device that has installed PhysioMeter can administer sessions without connectivity, but application updates only reach the device the next time it is online. Clinics that want a controlled update cadence should include that reconnect step in the device's maintenance routine.

Commercial-licensing constraints.

Many widely used physical therapy outcome measures, including a number of patient-reported instruments, are subject to copyright or commercial licensing. PhysioMeter's configuration-

driven design makes it straightforward to add such measures to a local deployment, but redistributing them within the open-source repository, or within a public preset, requires the appropriate licenses. Clinics integrating proprietary measures into their own deployments are responsible for securing those licenses.

No formal usability study.

The application was not evaluated through a formal usability study with a representative sample of physical therapists. The clinical review conducted during development (Section 5.2) supports clinical accuracy and workflow fit but does not yield generalizable usability data. A formal study using the System Usability Scale or similar instruments with multiple clinicians across different practice settings would strengthen claims about usability.

6.3 Future work

Expanded measure library.

Adding outcome measures for populations beyond geriatric mobility assessment is the most direct extension, with neuro-rehabilitation, patient-reported outcome measures, and measures for pediatric, vestibular, and cardiopulmonary populations as natural next targets.

Community contribution workflow.

A community process for contributing new outcome measures, including clinical review and validation procedures, would expand the measure library faster than single-author additions. This could follow an open-source contribution model with clinical peer review for interpretation rules and threshold data.

Longitudinal analysis.

The session model already supports multiple sessions per patient over time. Extending the interpretation engine to compare current results against previous sessions would support progress tracking.

Formal usability evaluation.

A formal usability study with a representative sample of physical therapists across multiple practice settings would generate generalizable usability data and identify interface improvements. Established instruments such as the System Usability Scale [6], together with multimethod usability assessments recommended for rehabilitation apps [3], provide a basis for that evaluation.

Multi-language support.

Administration instructions and user interface text are currently English-only. Adding support for multiple languages would let the tool be used in non-English clinical settings.

Graphical configuration editor.

YAML editing does not require programming, but clinicians may not be familiar with the syntax. A graphical user interface for editing the configuration files would lower the barrier further, enabling clinicians and researchers with limited computational background to contribute updates.

6.4 Dissemination

PhysioMeter is deployed as a static website at https://niema-lab.github.io/PhysioMeter/ and is freely accessible without account creation or subscription. The source code is available at https://github.com/Niema-Lab/PhysioMeter under the GNU General Public License, version 3 (GPL-3.0). A manuscript describing the tool is being prepared for submission to *Physical Therapy* (PTJ), the journal of the American Physical Therapy Association, as a Perspective article.

Chapter 6, in part, is currently being prepared for submission for publication of the material. Ji, Daniel; George-Moshiri, Karen; Puthoff, Michael; Flynn, Sheryl; Moshiri, Niema. The thesis author was the primary investigator and author of this material.

Chapter 7

Conclusion

This thesis presented PhysioMeter, a free, open-source, client-side web application for scoring and interpreting physical therapy outcome measures. Developed in collaboration with licensed physical therapists, it implements a declarative Measurements, Calculations, and Interpretations pipeline defined in YAML configuration files, runs entirely in the browser with no server-side processing of patient data, and is installable as a Progressive Web App for offline use at the point of care.

The primary contributions of this work are:

1. A declarative pipeline for outcome measure scoring in which the Measurements, Calculations, and Interpretations stages separate clinical logic from application code, letting clinical experts define, review, and modify scoring rules directly in YAML.

2. Privacy by architecture: the application runs entirely in the browser and encrypts records on disk under a clinician-set session password, avoiding the centralized-server breaches that account for most healthcare data incidents.

3. Extensibility through configuration, so that adding a new outcome measure requires only editing YAML files; configuration errors are caught automatically by the validation pipeline (Section 4.10) before deployment.

4. A working, freely accessible clinical tool that implements a complete assessment protocol (Annual Mobility Assessment), integrates the instruments a clinician needs during administration (stopwatches, timers, counters), supports patient and session management across visits, and exports results as PDF or CSV.

Together, these contributions make PhysioMeter the first outcome-measure scoring tool to combine client-side execution, clinician-configurable measure definitions, automated interpretation with normative data comparison, and free open-source availability.

Appendix A

Supported Outcome Measures

Table A.1 lists all outcome measures currently configured in PhysioMeter, including their input type, validation rules, and associated calculations and interpretations. The Vital Signs measurement is broken out into its three component fields (Resting Pulse Rate, Blood Pressure, Oxygen Saturation) for reference; the Annual Mobility Assessment preset (Table 3.2) administers Vital Signs as a single grouped step.

Table A.2 lists all calculations derived from the measurements above.

Table A.3 lists all interpretation rule chains, their trigger conditions, and severity levels.

Table A.1. Outcome measures configured in PhysioMeter.

Measure	Input Type	Trials	Validation
Resting Pulse Rate	Integer	—	0–300 bpm, integer
Blood Pressure	Text (pattern)	—	XX/YY format, systolic 0–200, diastolic 0–200, systolic $>$ diastolic
Oxygen Saturation	Decimal	—	0–100%
Vital Signs	Grouped fields	—	Combines the above three measurements
5-Meter Usual Walking Speed	Stopwatch	2	$>$0, $<$3600 seconds
5-Meter Fast Walking Speed	Stopwatch	2	$>$0, $<$3600 seconds
30-Second Sit to Stand	Countdown + Integer	—	Count: 0–100, integer
Assistive Device	Radio	—	None, Straight Cane, Other
Four Square Step Test	Stopwatch	2	$>$0, $<$3600 seconds
Modified Four Square Step Test	Stopwatch	2	$>$0, $<$3600 seconds
Timed Up and Go	Stopwatch	2	$>$0, $<$3600 seconds
TUG Cognitive Dual Task	Stopwatch + Integer	—	Time: $>$0, $<$3600s; Errors: 0–50, integer

Table A.2. Calculations configured in PhysioMeter.

Calculation	Operation	Source	Unit	Description
Age	age_from_date_of_birth	Date of Birth	years	Current age in complete years
Usual Walk Speed Mean	trial_mean	5-Meter Usual	seconds	Mean of 2 trials
Fast Walk Speed Best	trial_min	5-Meter Fast	seconds	Best (fastest) of 2 trials
30STS Count	field_min	30STS	repetitions	Repetition count
FSST Best	trial_min	FSST	seconds	Best of 2 trials
Modified FSST Best	trial_min	Mod. FSST	seconds	Best of 2 trials
TUG Best	trial_min	TUG	seconds	Best of 2 trials
TUG Cognitive Best	field_min	TUG Cognitive	seconds	Best time across instances

Table A.3. Interpretation rule chains configured in PhysioMeter. Numeric thresholds and age/sex-stratified normative data are taken from the APTA Geriatrics Annual Mobility Assessment Interpretation Charts [1].

Interpretation	Rules (summary)	Severities
Vital Signs	Ineligible if pulse >100, SpO_2 <90%, systolic >180 or <90, diastolic >110 or <60. Otherwise eligible.	concern, normal
Usual Walking Speed	Fall risk <0.76 m/s; frailty risk <0.63 m/s; age/sex zone threshold lookup.	concern, caution, normal
Fast Walking Speed	Fall risk <1.10 m/s; age/sex zone threshold lookup.	concern, caution, normal
30 Second Sit to Stand	Age/sex zone threshold lookup (SD-derived bands).	concern, caution, normal
Four Square Step Test	Multiple fall risk ≥15s; age/sex zone threshold lookup.	concern, caution, normal
Modified Four Square Step Test	Notes that the modified variant has no published normative thresholds; interpretation is based on test completion and clinician judgment.	info
TUG	Fall risk >12.0s (sex-specific Sn/Sp); frailty ≥17.8s; age/sex zone lookup.	concern, caution, normal
TUG Cognitive	Fall risk >11s; age/sex zone threshold lookup.	concern, caution, normal
Annual Mobility Assessment	Checks vital signs interpretation; if ineligible, warns to refer patient instead of proceeding.	caution, normal

Appendix B

In-App User Guide and Safety Notes

The end-user guide is bundled directly into the deployed application and is accessible from the home page at the `/user-guide` route (Figure B.1). It covers setting and using the session password (first-time setup, unlock, manual lock, the thirty-minute inactivity auto-lock, and the typed-confirmation reset), installing PhysioMeter as a Progressive Web App, creating a patient, starting a session, administering measurements, and reviewing the summary. The same content lives in `docs/user-guide.md`; a Playwright test (`e2e/user-guide-walkthrough.spec.js`) regenerates the screenshots and step text against the live UI.

Safety notice.

PhysioMeter is a workflow and scoring aid, not a diagnostic device. Interpretations are derived from published thresholds and normative data; clinical judgment is required for all decisions. PhysioMeter must not be used outside the scope of practice of a licensed clinician.

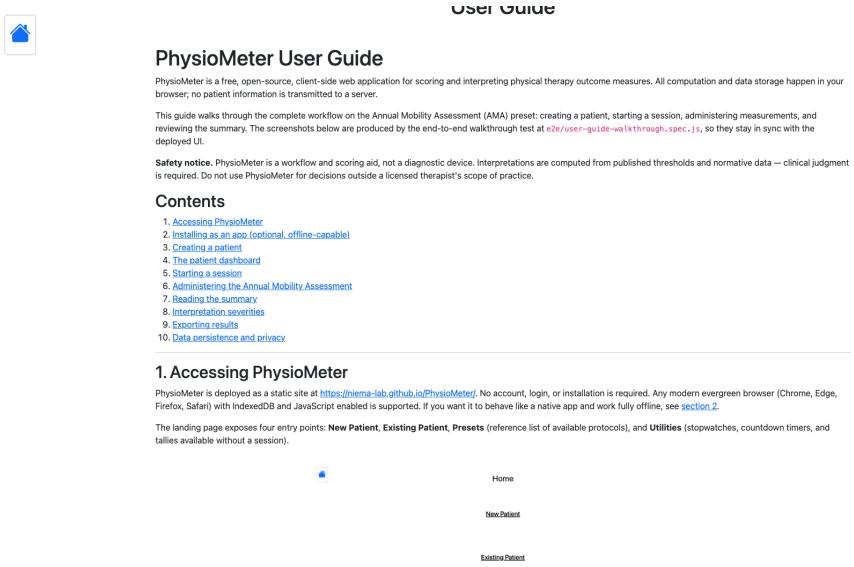

Figure B.1. The in-app user guide page at `/user-guide`, showing the table of contents and opening section.

Bibliography

[1] APTA Geriatrics. Annual mobility assessment interpretation charts, version 2.1. https://aptageriatrics.org/wp-content/uploads/2026/02/AMA-InterpChartsFinalv2.1.pdf, 2026. Accessed 2026-04-17.

[2] Douglas P Gross, Susan Armijo-Olivo, William S Shaw, Kelly Williams-Whitt, Nicola T Shaw, Jan Hartvigsen, Ziling Qin, Christine Ha, Linda J Woodhouse, and Ivan A Steenstra. Clinical decision support tools for selecting interventions for patients with disabling musculoskeletal disorders: A scoping review. *Journal of Occupational Rehabilitation*, 26(3):286–318, 2016.

[3] Sylvia Hach, Gemma Alder, Verna Stavric, Denise Taylor, and Nada Signal. Usability assessment methods for mobile apps for physical rehabilitation: Umbrella review. *JMIR mHealth and uHealth*, 12:e49449, 2024.

[4] Health Level Seven International. Clinical quality language (CQL) specification, release 1. https://cql.hl7.org/, 2019. HL7 Standard. Accessed 2026-04-16.

[5] HIPAA Journal. Healthcare data breach statistics. https://www.hipaajournal.com/healthcare-data-breach-statistics/, 2026. Accessed 2026-04-16.

[6] Maciej Hyzy, Raymond Bond, Maurice Mulvenna, Lu Bai, Alan Dix, Simon Leigh, and Sophie Hunt. System usability scale benchmarking for digital health apps: Meta-analysis. *JMIR mHealth and uHealth*, 10(8):e37290, 2022.

[7] Diane U Jette, James Halbert, Courtney Iverson, Erin Miceli, and Palak Shah. Use of standardized outcome measures in physical therapist practice: Perceptions and applications. *Physical Therapy*, 89(2):125–135, 2009.

[8] Anat Kristal, Ignacio A Gaunaurd, Sara J Morgan, Geoffrey S Balkman, Rachael E Rosen, Rana Salem, Alyssa M Bamer, and Brian J Hafner. Use of standardized outcome measures among physical therapists in the United States: A cross-sectional survey study. *PLOS ONE*, 20(8):e0330528, 2025.

[9] Brian McDonnell, Shannon Stillwell, Shelby Hart, and Roger B Davis. Breaking down barriers to the utilization of standardized tests and outcome measures in acute care physical therapist practice: An observational longitudinal study. *Physical Therapy*, 98(6):528–538, 2018.

[10] Jill Meirte, Nick Hellemans, Mieke Anthonissen, Lenie Denteneer, Koen Maertens, Peter Moortgat, and Ulrike Van Daele. Benefits and disadvantages of electronic patient-reported outcome measures: Systematic review. *JMIR Perioperative Medicine*, 3(1):e15588, 2020.

[11] Matthias Samwald, Karsten Fehre, Jeroen de Bruin, and Klaus-Peter Adlassnig. The Arden Syntax standard for clinical decision support: Experiences and directions. *Journal of Biomedical Informatics*, 45(4):711–718, 2012.

[12] Nestor Cavalcante Teixeira Neto, Yuri Lopes Lima, Gabriel Peixoto Leão Almeida, Márcio Almeida Bezerra, Pedro Olavo De Paula Lima, and Rodrigo Ribeiro de Oliveira. Physiotherapy questionnaires app to deliver main musculoskeletal assessment questionnaires: Development and validation study. *JMIR Rehabilitation and Assistive Technology*, 5(1):e1, 2018.

[13] U.S. Department of Health and Human Services, Office for Civil Rights. Breach portal: Notice to the secretary of HHS breach of unsecured protected health information. https://ocrportal.hhs.gov/ocr/breach/breach_report.jsf, 2026. Accessed 2026-04-16.