

Electronic Health Records System for PIES Fitness Yoga Studio

Members:

Devon Boldt, Brett A. Noneman, Charles P. Wiecking, James C. Yeh

Product Description:

Developed in collaboration with PIES Fitness Yoga, the PIES Yoga Therapy Electronic Health Records (EHR) system is a secure, role-based web application that enables therapists to efficiently manage client documentation, scheduling, and session workflows. Junior therapists can create and submit Intake, SOAP, and Self-Assessment forms, search and filter client records, review past notes, flag sessions for follow-up, and receive confirmation upon successful submission. Senior therapists have all junior therapist capabilities plus organization-wide visibility, advanced filtering by therapist name, client name, date, and session type, and the ability to export client summaries and monitor activity.

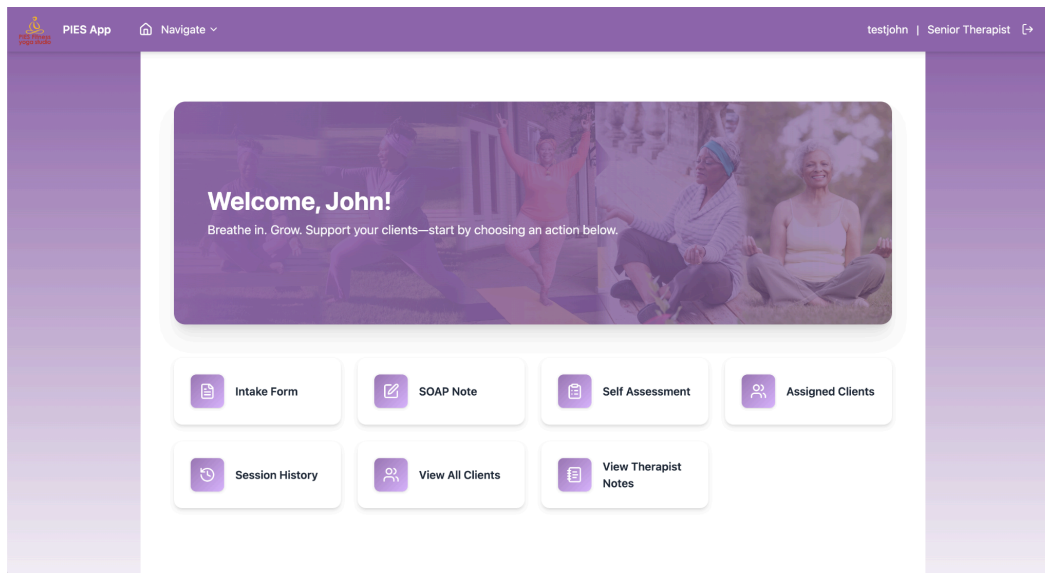
The system features a React-based frontend, a Spring Boot backend, and a MySQL database, connected through REST APIs secured with JWT authentication. Core infrastructure includes Docker-based deployment, Swagger/OpenAPI documentation, and Flyway-managed schema migrations. All data is encrypted in transit and at rest, with strict role-based permissions to protect privacy and confidentiality. The final implementation supports appointment scheduling, secure document viewing and downloading, and an enhanced user interface refined through iterative feedback from senior therapists and administrative staff. By project completion, the platform met all defined functional requirements, reduced administrative overhead, improved data accuracy, and supported more personalized client care.

Functionalities:

Dashboard:

The dashboard serves as the central navigation hub for therapists, providing a role-aware interface that adapts to each user's permissions. Junior therapists have quick access to their assigned clients, active sessions, and documentation forms. Senior therapists have expanded access to organization-wide client records and administrative tools. The page prominently displays the therapist's name and role, along with pending tasks, upcoming appointments, and quick actions. A consistent navigation bar allows smooth transitions to all core system pages.

During later sprints, the dashboard design was enhanced with improved images, refined layouts, and responsive navigation. Functional elements include direct entry points to Intake, SOAP, and Self-Assessment forms, as well as search and filtering capabilities for client records. For senior therapists, advanced filters by therapist name, session type, and date enable oversight of clinical activity. By consolidating high-priority tools and information in a single location, the dashboard streamlines navigation and reduces the number of steps required to complete common tasks.



Intake Form:

The intake form page captures a client's initial health and wellness profile before therapy sessions begin. It includes sections for demographic details, medical history, yoga experience, and current health conditions, with dynamic checkbox groups to streamline data entry. Conditional fields appear based on prior selections, such as additional medication details if the client reports taking any. A digital signature field records client acknowledgment directly within the form.

By Sprint 3, the intake form was fully integrated with the backend, enabling data validation, secure storage, and persistence. Autofill features reduce duplicate entry, particularly when transitioning to SOAP notes. Error handling provides immediate feedback if required fields are missing or entered incorrectly, ensuring accurate data capture. The form's structure and validation logic align with clinical documentation standards, establishing a reliable starting point for each client's record in the system.

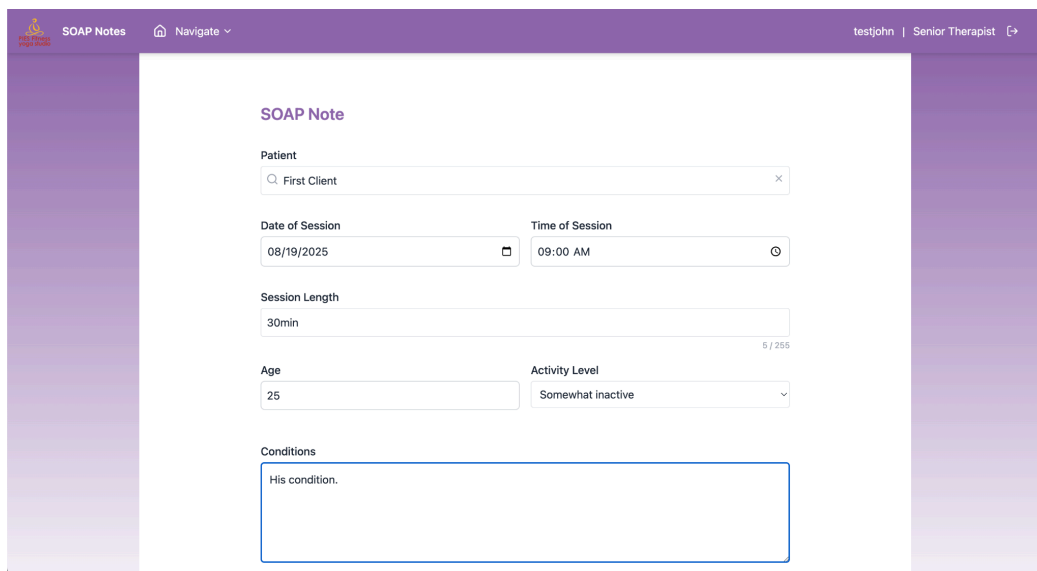
The screenshot displays the 'Intake Form' interface. At the top, there is a navigation bar with 'Intake Form' and 'Navigate' on the left, and 'testjohn | Senior Therapist' on the right. The main content area is titled 'Client' and includes a dropdown menu for 'Using existing client: Second Client (#2)' with a 'Clear / New client' link. Below this is a section for 'Assign Therapist (from client record)' with a dropdown menu showing 'Example Therapist'. The 'Confidential Information' section contains several input fields: 'First Name' (Second), 'Last Name' (Client), 'Date of Birth' (01/01/1980), 'Address' (6 Drive), 'City' (Charlotte), and 'State' (North Carolina). Each field has a character count indicator (e.g., 6 / 100).

The screenshot displays the 'Intake Form' interface, focusing on the 'Goals / Expectations' and 'Personal Yoga Interests' sections. The 'Goals / Expectations' section includes checkboxes for 'Improve fitness', 'Injury rehabilitation', 'Strength training', 'Other', 'Increase well-being', 'Positive reinforcement', and 'Weight management' (checked). The 'Personal Yoga Interests' section includes checkboxes for 'Asana (postures)', 'Meditation', 'Eastern energy systems', 'Pranayama (breath work)', 'Yoga Philosophy', and 'Other'. The 'Lifestyle & Fitness' section includes a dropdown menu for 'Current activity level' set to 'Somewhat active'.

SOAP Note:

The SOAP note page enables therapists to document client progress using the structured Subjective, Objective, Assessment, and Plan format. Features include patient lookup, therapist auto-linking, and autofill of relevant data from the intake form. Quick notes can be added outside the structured sections for informal observations, and the goals field allows therapists to record follow-up actions or homework assignments.

Integration with the backend ensures that each SOAP note is linked to the correct client record and session, with persistence for future review. By Sprint 4, the system allowed viewing of a client's previous SOAP notes directly from this page, enhancing continuity of care. User interface refinements focused on clarity, with clearly separated sections, date-stamping, and simple navigation between current and past notes. The form follows best practices for therapeutic documentation while providing flexibility for individualized treatment planning.



The screenshot displays a web application interface for creating a SOAP note. The header includes the 'SOAP Notes' title, a 'Navigate' dropdown menu, and the user's name 'testjohn | Senior Therapist' with an edit icon. The main content area is titled 'SOAP Note' and contains several input fields: a 'Patient' dropdown menu with 'First Client' selected; 'Date of Session' (08/19/2025) and 'Time of Session' (09:00 AM) fields; a 'Session Length' field (30min) with a '5 / 255' character count; 'Age' (25) and 'Activity Level' (Somewhat inactive) fields; and a 'Conditions' text area containing the text 'His condition.' The interface is clean and modern, with a purple header and a white main area.

Self Assessment:

The self-assessment page allows therapists to evaluate their own performance, session quality, and client engagement. It mirrors the structure of other forms in the system, combining text inputs, checkboxes, and digital signature capture. This form supports reflective practice, encouraging self-awareness and ongoing professional growth.

Originally a static interface in early sprints, the self-assessment page was fully integrated with the backend by Sprint 4, including validation and secure data persistence. Submission confirmation alerts provide immediate feedback, and completed forms are stored for review by senior therapists. This feature contributes to quality assurance and supports organizational oversight by enabling supervisors to track therapist development trends over time.

The image displays two screenshots of a self-assessment form. The top screenshot shows the form's header and several text input fields. The bottom screenshot shows a list of checkboxes for various assessment categories.

Form Header: Self Assessment | Navigate | testjohn | Senior Therapist

Form Fields:

- Patient: Fourth Client
- Date of Session: 08/04/2025
- How did the client(s) react to the tools presented?: His react.
- How did the client(s) react to you?: Another react.
- How did you respond to the client(s)? My respond.
- What adaptations and/or modifications did you utilize?

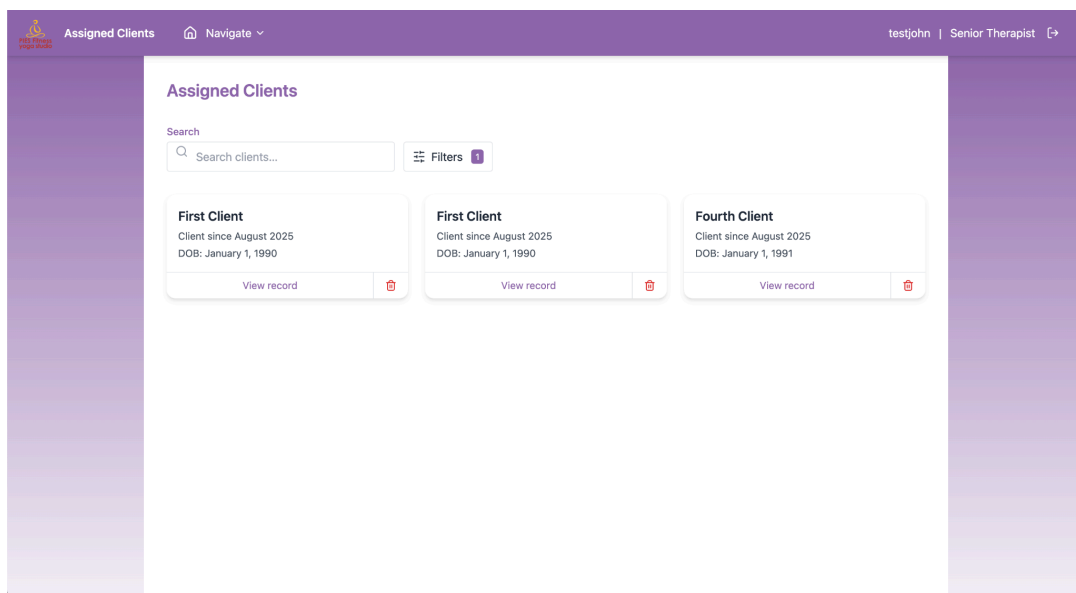
Form Categories and Checkboxes:

- Physical:** Physical
- Emotional:** Emotional
- Intellectual:** Intellectual
- Spiritual:** Spiritual
- Asana:**
 - Sun A
 - Seated
 - Prone
 - Balance
 - Backbends
 - Sun B
 - Standing
 - Supine
 - Revolved
 - Inversion
- Mindfulness:**
 - Guided Visualization
 - Internal Observation
 - Non-judgement
 - Breath-centered
 - External Observation
 - Other
- Kleshas:**
 - Ignorance
 - Attachment
 - Fear of Loss
 - Egoism
 - Aversion
- Chakras:**
 - Root
 - Solar Plexus
 - Throat
 - Crown
 - Sacral
 - Heart
 - Third-Eye

Assigned Clients

The assigned clients page displays all clients linked to the currently logged-in therapist. It includes search and filtering tools to quickly locate clients by name or session date, along with indicators for active and inactive clients. Each entry provides key details such as last session date, upcoming appointments, and quick links to relevant forms and session history.

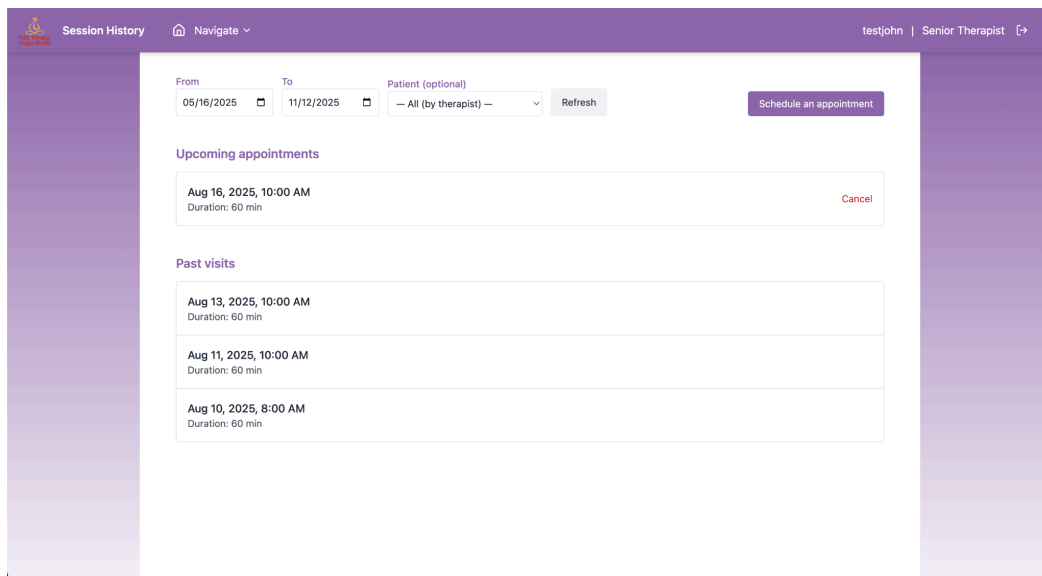
The design prioritizes efficiency, allowing therapists to manage their caseload from a single view. Soft delete functionality enables hiding inactive clients without removing their records from the database. For junior therapists, this page is central to daily workflow management, while senior therapists use a parallel “View All Clients” page for organization-wide oversight.



Session History

The session history page provides a chronological record of all documented sessions for a specific client. It displays summaries of each session's SOAP notes, intake updates, and any associated self-assessments. Therapists can filter sessions by date or type, such as asana or meditation, to focus on relevant periods of treatment.

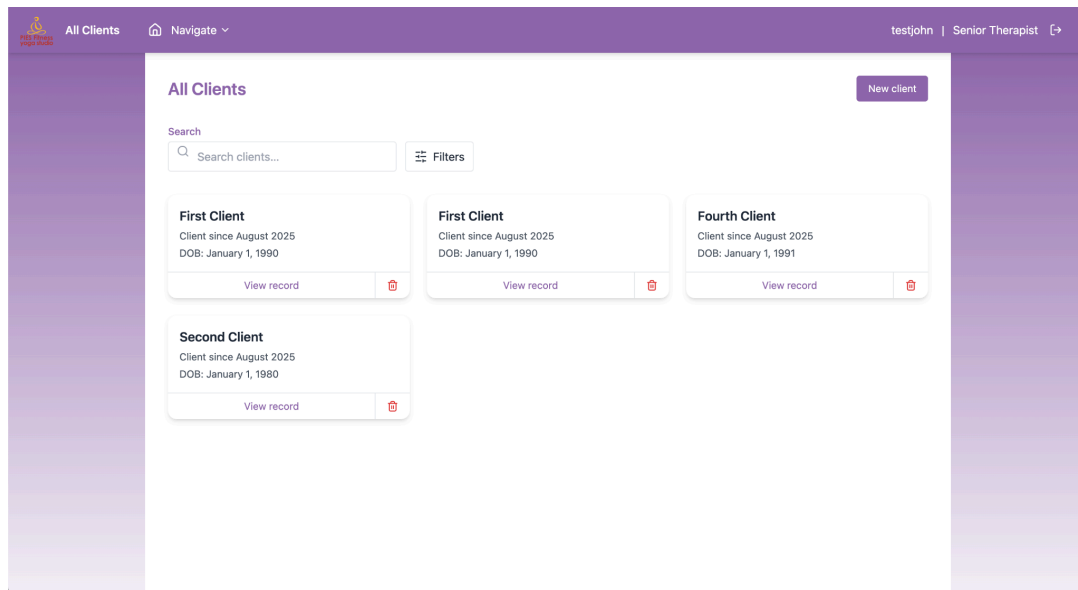
In Sprint 4, the page was enhanced with follow-up tagging, allowing therapists to mark sessions that require continued attention. Selecting a session entry opens the full documentation for review or editing. By providing a comprehensive view of client progress over time and ensuring persistent backend storage, the session history page supports clinical continuity and informed treatment planning.



View All Clients

The View All Clients page provides senior therapists with an organization-wide view of all registered clients. Advanced filtering options by therapist name, client name, session date, and session type allow supervisors to quickly locate records for case reviews, quality audits, or oversight of junior therapist workloads.

In addition to search and filter functions, the page supports export capabilities for client summaries, enabling the generation of PDFs or other report formats. The layout mirrors the Assigned Clients page for consistency but includes expanded access privileges. This feature plays a key role in organizational management by ensuring that all client care is tracked, documented, and reviewable in accordance with practice standards.



Design

Overview

The PIES Yoga Therapy EHR system design focuses on modularity, scalability, and secure role-based access. The architecture integrates clear role definitions, a layered backend structure, and robust deployment practices to ensure maintainability and consistent performance. The following subsections outline the system's core design elements, supported by diagrams that illustrate use cases, deployment flow, backend module organization, and database relationships.

Use Case Diagram (*Figure 1*)

The use case diagram illustrates how users interact with the platform, highlighting available actions, enforced processes, and relationships between use cases. Three main actors are identified: Client (Patient), Therapist, and Senior Therapist.

Clients primarily provide intake information, either written or typed, which therapists enter into the system. Therapists are the primary operational users, performing tasks such as logging in, entering intake data, creating SOAP notes and self-assessments, viewing assigned clients, searching client records, reviewing session history, and viewing therapist notes. Senior therapists inherit all therapist capabilities while adding administrative functions, including managing junior therapist accounts.

The diagram uses <<include>> relationships for mandatory steps, such as role-based access enforcement after login or field validation in forms. <<extend>> relationships represent optional actions, such as assigning follow-up tasks after reviewing session history. Generalization links allow senior therapists to inherit all therapist use cases while adding administrative privileges.

This structure enforces data integrity and accountability through validation, timestamping, and change logging. Future enhancements could include expanded client engagement features, more explicit extension conditions, and additional security-focused use cases like password resets or explicit logout.

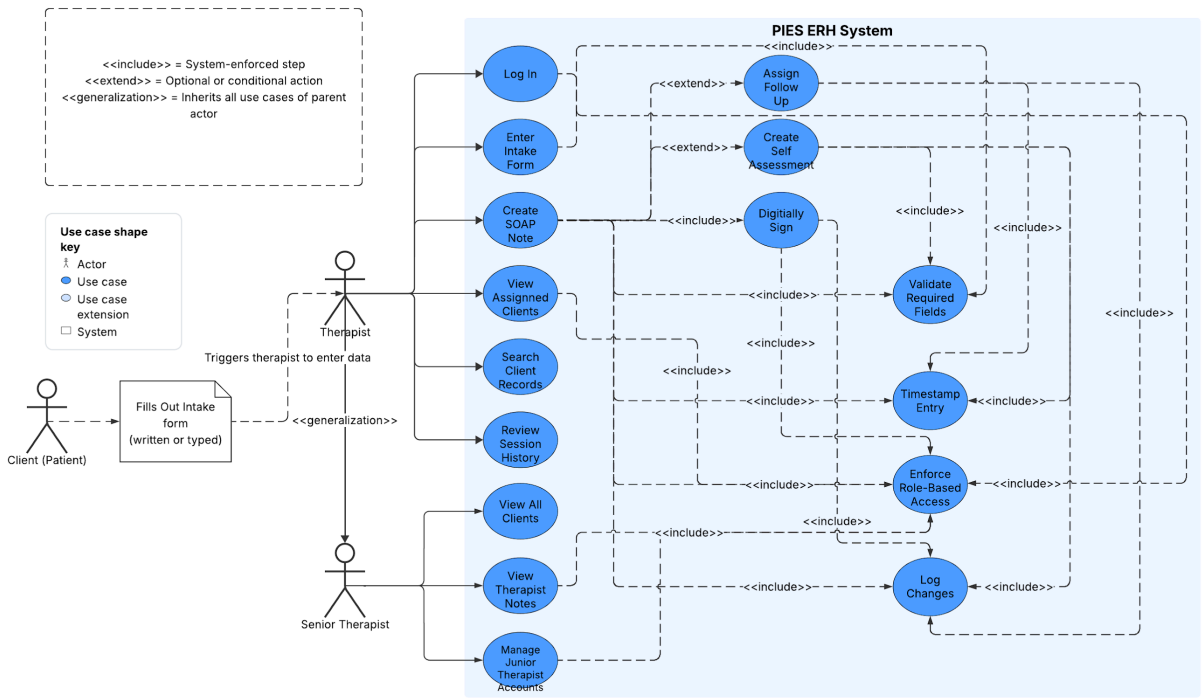


Figure 1. Use Case Diagram showing role-specific actions, relationships, and system-enforced processes.

Deployment Architecture (Figure 2)

The deployment diagram outlines how user requests are processed and how system components interact at runtime. Clients connect via web or mobile interfaces over HTTPS. Requests first pass through Cloudflare, serving as a CDN, proxy, and web application firewall for performance optimization and security filtering. Traffic then proceeds to the pies-her-api service, a Spring Boot application running inside a Docker container on a Google Cloud Platform (GCP) virtual machine.

A CI/CD pipeline using GitHub Actions automates deployment, pushing updated application builds into the environment. Docker Compose orchestrates both the API and MySQL containers, with configuration managed through .env, application.yml, and logback-spring.xml for consistent setup and logging.

On startup, Flyway migrations ensure the MySQL schema matches the latest application requirements. The application communicates with MySQL over JDBC and exports logs to GCP services such as Cloud Logging and Stackdriver for centralized monitoring. This architecture combines secure traffic handling, automated migration, and scalable containerized deployment.

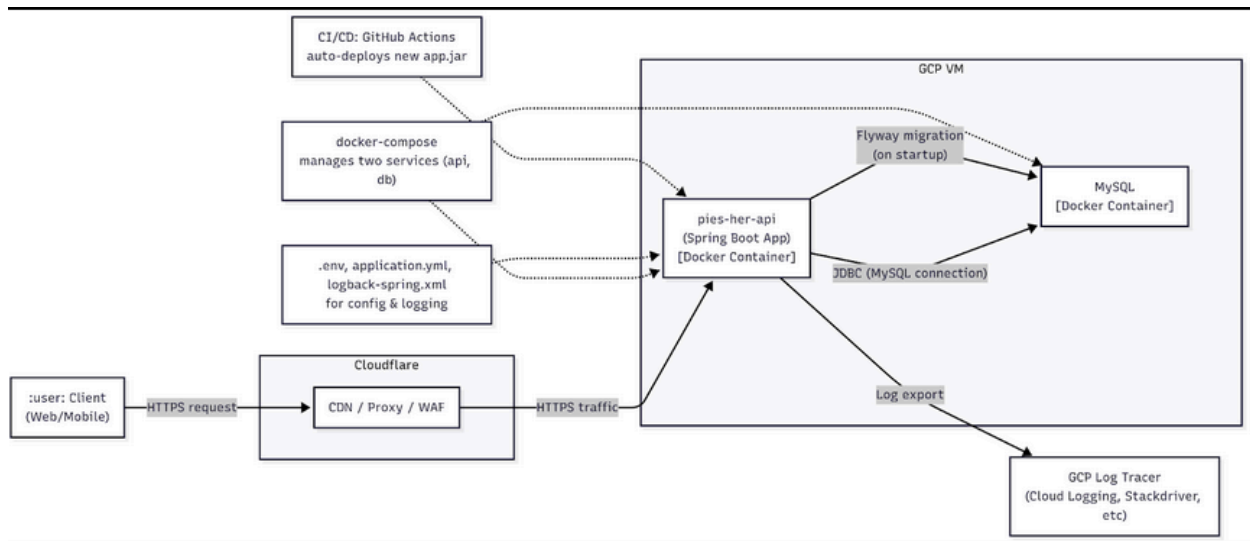


Figure 2. Deployment Architecture showing request flow, infrastructure components, and container orchestration.

Backend Module Architecture (Figure 3)

The backend architecture organizes the pies-her-api application into modular, feature-specific layers. At the core is the main application module, which coordinates all features and services. A Shared Code module provides common utilities and Data Transfer Objects (DTOs), reducing duplication and promoting consistency.

Each feature module follows a layered design:

- **Controllers** manage incoming API requests
- **Services** implement business logic
- **Repositories** handle database interactions

Therapist, Patient, and Intake Form modules each follow this structure. Security and authentication are implemented in a dedicated module containing authentication controllers, a JWT service, and security configuration classes. Application settings are managed through configuration files, including application.yml for runtime settings, .env for environment variables, and logback-spring.xml for logging.

Database interactions occur through MySQL, with Flyway migrations keeping schema updates synchronized with code changes. The modular architecture promotes maintainability, scalability, and security while ensuring consistent development practices across modules.

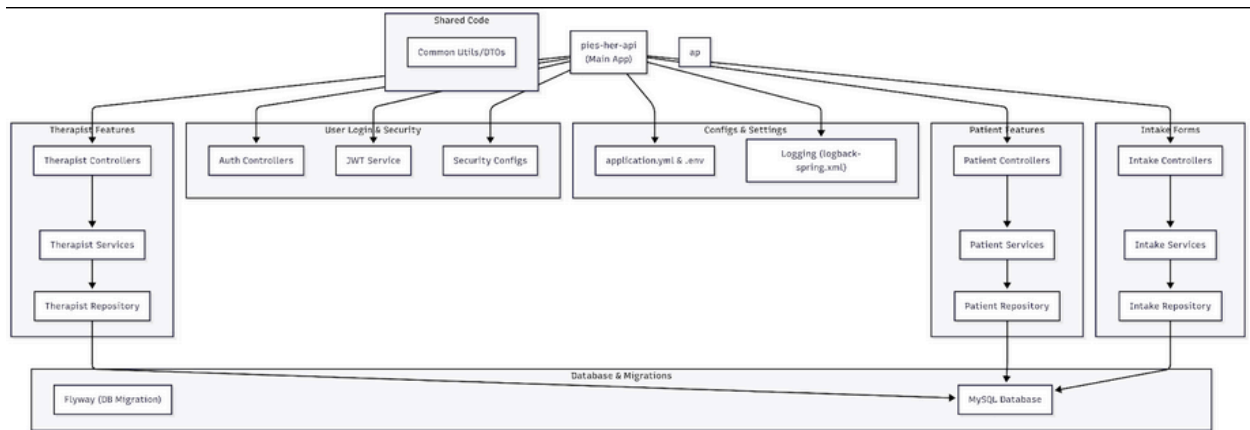


Figure 3. Backend Module Architecture showing feature modules, shared code, configuration, and database integration.

Entity Relationship Diagram (*Figure 4*)

The Entity Relationship Diagram (ERD) defines the database schema for the PIES Yoga Therapy EHR system, capturing how data is organized, linked, and maintained. The design enforces referential integrity through foreign key constraints and supports system functionality through well-structured relationships.

Core Entities and Relationships

- **therapists** – Stores personal information, authentication credentials, role designation, and account status. Each therapist can be assigned multiple patients
- **patients** – Contains demographic and contact details, emergency contact information, and links to assigned therapists. Each patient can have multiple intake forms, SOAP notes, and self-assessments
- **intake_forms** – Records a client's initial health and wellness profile. Linked to both *patients* and *therapists*
- **soap_notes** – Stores structured session documentation (Subjective, Objective, Assessment, Plan) along with session details, linked to both *patients* and *therapists*
- **self_assessments** – Captures a therapist's evaluation of their own session performance, linked to both *patients* and *therapists*
- **intake_form_health_history** – Represents detailed medical history and health conditions tied to a specific intake form
- **audit_logs** – Tracks system events with details such as username, action, entity affected, and timestamp
- **flyway_schema_history** – Maintains version control for database schema changes
- **installed_ran** – Tracks installed versions and execution details for database updates.

Design Characteristics

- All primary entities are linked using foreign keys, ensuring data consistency
- Indexed fields, such as IDs and frequently queried attributes (e.g., patient names, therapist IDs, session dates), improve query performance
- The *intake_form_health_history* table normalizes condition tracking to avoid redundancy in the main intake form table
- Audit logging supports traceability and accountability across the system
- Flyway migrations manage schema versioning to synchronize database structure with application changes

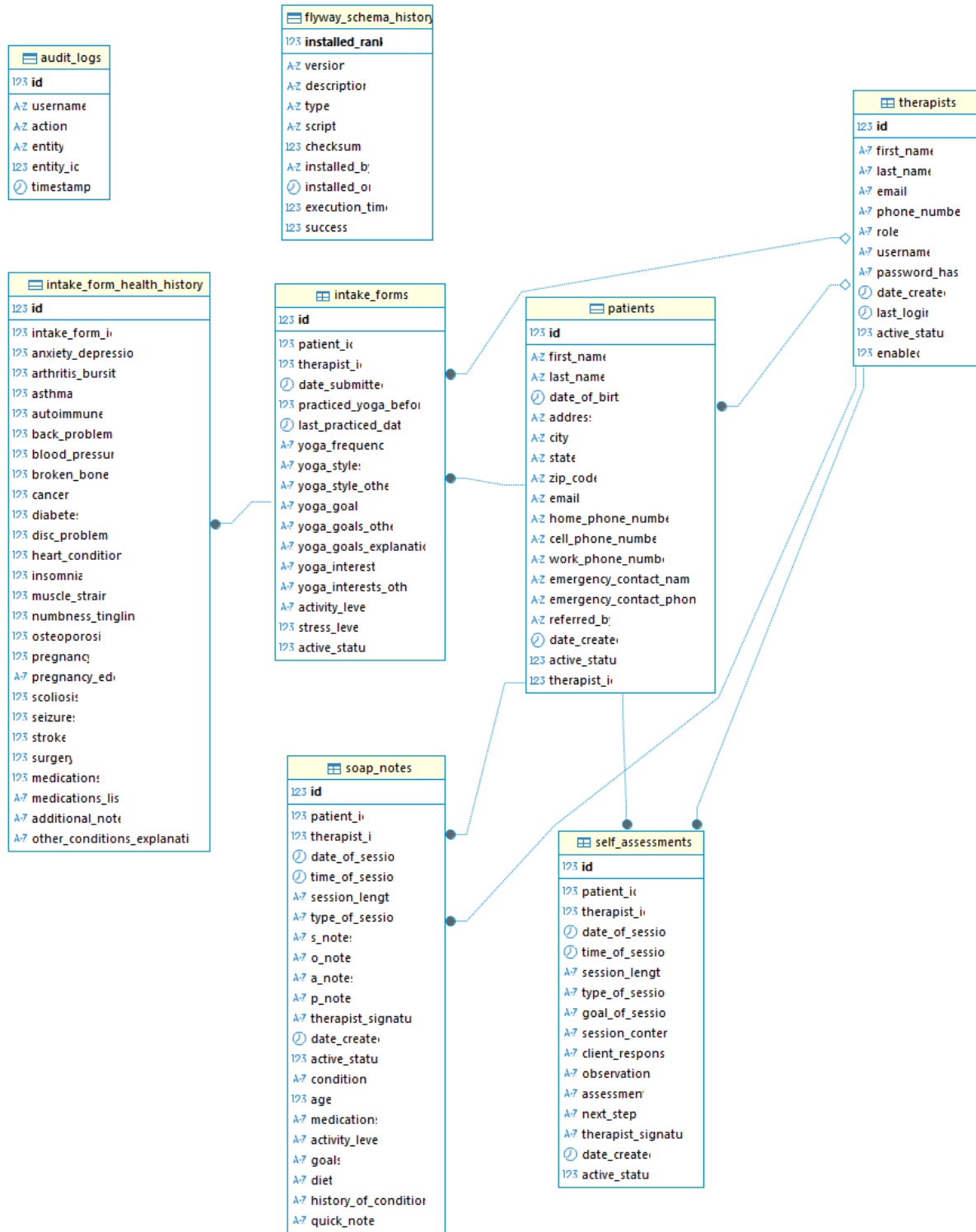


Figure 4. Entity Relationship Diagram showing therapists, patients, forms, notes, assessments, audit logs, and schema management tables with their relationships.

API Design

The PIES Yoga Therapy EHR exposes a REST API documented with OpenAPI 3.1. JSON is used for requests and responses. Authentication uses JWT. Role-based access control limits read and write operations by role and assignment.

Authentication

Method	Path	Purpose
POST	/auth/register	Create therapist account (admin or seeded only)
POST	/auth/login	Authenticate and issue JWT
GET	/auth/me	Return current therapist profile from token

Therapists

Method	Path	Purpose
GET	/therapists	List therapists with pagination
GET	/therapists/active	List active therapists
GET	/therapists/{id}	Fetch therapist by id
POST	/therapists	Create therapist
PUT	/therapists/{id}	Update therapist
DELETE	/therapists/{id}	Deactivate or remove per policy

Patients

Method	Path	Purpose
GET	/patients	Search and list patients (paged)
GET	/patients/{id}	Get patient details
POST	/patients	Create patient and assign therapist
PUT	/patients/{id}	Update patient
DELETE	/patients/{id}	Soft delete or deactivate

Intake Forms

Method	Path	Purpose
GET	/intakes	List intake forms (paged)
GET	/intakes/{id}	Get intake form by id
GET	/intakes/patient/{patientId}	List a patient's intake forms
POST	/intakes	Create intake form
PUT	/intakes/{id}	Update intake form
DELETE	/intakes/{id}	Delete per retention policy

SOAP Notes

Method	Path	Purpose
GET	/soap-notes	List SOAP notes (paged)
GET	/soap-notes/{id}	Get SOAP note
POST	/soap-notes	Create SOAP note
PUT	/soap-notes/{id}	Update SOAP note
DELETE	/soap-notes/{id}	Delete per retention policy

Self Assessments

Method	Path	Purpose
GET	/self-assessments	List self assessments (paged)
GET	/self-assessments/{id}	Get self assessment
POST	/self-assessments	Create self assessment
PUT	/self-assessments/{id}	Update self assessment
DELETE	/self-assessments/{id}	Delete per retention policy

Appointments

Method	Path	Purpose
POST	/appointments	Create appointment
GET	/appointments/therapist/{therapistId}	List therapist appointments
GET	/appointments/patient/{patientId}	List patient appointments
DELETE	/appointments/{id}	Cancel appointment

Common behaviors

- **Auth header:** Authorization: Bearer <token>
- **RBAC:** Senior therapists may read across the organization. Junior therapists are scoped to assigned patients
- **Pagination:** List routes accept page, size, and sort
- **Validation:** DTO constraints return 400 with ApiError
- **Audit:** Mutations are logged with user, entity, action, timestamp
- **Soft delete:** Deactivated records are hidden by default queries, not hard removed

Representative payloads

- **Login request**

```
{ "username": "alice", "password": "secret" }
```

- **Login response**

```
{ "accessToken": "<jwt>", "tokenType": "Bearer", "expiresIn": 3600 }
```

- **Error envelope (ApiError)**

```
{  
  "status": 400,  
  "error": "Bad Request",  
  "messages": ["field 'patientId' is required"],  
  "timestamp": "2025-08-15T17:05:00Z"  
}
```

Retrospective

What We Accomplished During the Sprints

Over the course of Sprints 2 through 4, the team evolved the PIES EHR system from a partially functional prototype into a fully integrated, role-based platform meeting all initial requirements.

In Sprint 2, core backend integration with MySQL was achieved, and therapist registration along with intake form submission was implemented end-to-end. The frontend intake form was developed with Tailwind CSS and react-hook-form, supporting dynamic fields, checkbox groups, and signature capture. This marked the first complete data submission from frontend to backend.

Sprint 3 expanded the system into full-stack functionality. The Intake and SOAP Notes forms were completed with conditional field rendering, checkbox serialization, real-time validation, and autofill between forms. Additional frontend pages for Assigned Clients, View All Clients, and Therapist Notes were built with role-based dashboards for junior and senior therapists. On the backend, full CRUD operations were implemented for therapists, patients, SOAP notes, self-assessments, and intake forms, with all endpoints protected by JWT authentication. Entity relationships were finalized in Spring Data JPA, robust error handling was added, and signature export logic was completed.

In Sprint 4, all remaining features were finalized. The self-assessment form was fully integrated with backend APIs, appointment scheduling was added, and document viewing/downloading was implemented with secure role-based access. UI enhancements incorporated stakeholder feedback, expanded search/filter capabilities were introduced, and the application was polished for presentation and deployment. Backend changes focused on final integrations and performance refinements.

Did We Implement All User Stories Specified for this Release?

By the conclusion of Sprint 4, all user stories defined at the start of the project were implemented. Junior therapists can submit intake, SOAP, and self-assessment forms; search clients by name, date, and session type; view past records; flag sessions for follow-up; and receive submission confirmations. Senior therapists have full junior capabilities plus organization-wide record visibility, advanced filtering, export options, and activity monitoring.

Stories carried over from earlier sprints, such as advanced filtering, follow-up tagging, self-assessment integration, and SOAP note history, were completed in Sprint 4. Additional enhancements like appointment scheduling, document access, and UI refinements were delivered as part of final scope.

What Impediments Did We Encounter?

The team faced both technical and process-related challenges throughout development:

- Sprint 2: Checkbox value mapping issues, incomplete frontend authentication flow, unvalidated signature capture, and missing conditional validation for certain fields
- Sprint 3: Complex checkbox serialization, race conditions in role-based rendering, incomplete signature submission pipeline, inconsistent field-level validation, backend DTO changes breaking frontend forms, and incomplete backend role parsing
- Sprint 4: Decentralized team schedules slowing integration, payload alignment for final features, state management complexity for new workflows, filter edge cases producing no results, and UI layout regressions from late-stage changes

How Did We Address Them?

Across all sprints, the team resolved blockers through incremental debugging, architecture refinements, and improved coordination:

- Added targeted logging and payload inspection in frontend forms to debug data mapping issues
- Created utility functions (`sanitizeKey`, `getSelectedOptions`) to align frontend payloads with backend DTOs
- Modularized form structures to reduce nesting complexity and improve maintainability
- Decoded JWTs client-side for early role-based rendering, eliminating undefined states on dashboard load
- Synchronized backend schema changes with parallel frontend updates to prevent contract mismatches
- Conducted structured integration testing for final features like self-assessment and document downloads
- Enhanced React state management for appointment scheduling and document viewing workflows
- Incrementally validated UI updates to prevent regression and maintain consistency
- Maintained issue tracking and sprint planning via GitHub, Trello, and Discord, adapting workflows to decentralized availability

What Went Well?

- Achieved full-stack integration for all major forms, with secure, role-based workflows
- Built reusable, modular UI components with Tailwind CSS and react-hook-form, enabling efficient form creation and validation
- Established a stable backend with secure authentication (BCrypt + JWT) and robust error handling
- Delivered responsive, role-aware dashboards and expanded client/session management tools
- Successfully completed all carried-over user stories and final scope enhancements
- Maintained effective coordination despite remote collaboration, leveraging GitHub and Discord for issue tracking and progress updates
- Incorporated stakeholder feedback into UI improvements without introducing regressions

What Could Be Improved?

- Begin integration testing earlier in each sprint to detect alignment issues before they block progress
- Implement validation rules during initial form development instead of deferring to later stages
- Improve estimation accuracy and encourage developer pairing for complex tasks
- Strengthen communication on API/schema changes to avoid unexpected frontend breakage
- Make project boards (Trello, GitHub) the single source of truth for task tracking and blocker status
- Standardize testing checklists for search and filtering features to catch edge cases early
- Integrate UI regression testing into the CI/CD pipeline to automatically flag layout or style inconsistencies
- Establish fixed check-in cadences to improve coordination during decentralized schedules