Project Management Plan

Healthsmart Medication Management Plan

Published on: 1/20/2025

Version: 1.1.1

Author: Joshua D’Agostino

Revision History			
Version	Author	Version Description	Date Completed
1.1.1	Joshua D’Agostino	Initial version of the document	1/20/2025

Table of Contents

1. Introduction

The Healthsmart Medication Management System is an innovative platform designed to assist patients in effectively managing their medications, thereby improving their health outcomes and overall quality of care. The system provides patients with a structured, user-friendly method for tracking their medications, dosage schedules, and ensuring timely reminders and updates to facilitate adherence to prescribed regimens. Additionally, it facilitates seamless communication between patients and healthcare providers, promoting continuous support and timely interventions.

For healthcare providers, the system offers a comprehensive suite of tools to effectively manage their patients' medications, monitor dosage schedules, and review refill requests. It also includes a direct communication channel that fosters better engagement between providers and patients. By automating routine tasks and simplifying medication management, the system reduces administrative burdens, allowing healthcare professionals to focus more on patient care. This streamlined approach ensures a higher standard of care while enhancing the efficiency and effectiveness of medication management.

1.1. Problem Statement

Medication management in the healthcare system faces significant challenges, many of which are influenced by external factors. A primary issue is medication non-adherence, where patients frequently struggle to follow prescribed dosage schedules. This non-adherence can stem from various factors, such as forgetfulness, lack of organization, or confusion about complex dosage instruction. In particular, the prescribing of multiple medications at once, often from different drug classes, can further compliance adherence, particularly for those without a healthcare background.

For healthcare providers, tracking medication adherence remains a significant challenge. It can be difficult for providers to consistently monitor whether patients are adhering to their prescribed regimens, and managing refill requests is often inefficient and prone to errors. Compounding these challenges is the patient's forgetfulness or lack of communication, making it difficult to determine if they are following their dosage schedules. This gap in adherence and communication has serious implications, particularly in the management of chronic diseases, where consistent medication adherence is crucial for effective disease control and preventing complications. Medication mismanagement has the potential to cause worsening health conditions, an increase in emergency visits, and, as a result, escalating healthcare costs for the patient.

2. Project Objectives

The Healthsmart Medication Management System has five objectives, which are achieved through its core functionality:

1. *Improve Medication Adherence*

This system will help patients track and adhere to their prescribed medication regimens. Features such as medication reminders, dosage alerts, and clear easy-to-follow schedules will enhance the likelihood that patients take their medications as prescribed.

2. *Support Healthcare Providers*

This system will equip clinicians with tools to monitor their patients' medication adherence, giving valuable insights into when interventions may be necessary. Healthcare providers will be able to adjust treatment plans, follow up on non-adherence cases, and ensure their patients receive the proper medications at the right time.

3. *Enhance Patient-Provider Communication*

An integrated communication channel will allow patients to exchange messages with their healthcare providers. This ensures that medication-related concerns are addressed in a timely manner, leading to better outcomes. It will also further facilitate effective collaboration between patients and their clinicians, further improving care coordination.

4. *Enable Easy Medication Management with Pharmacies*

The system will offer seamless integration with third-party pharmacy services, allowing patients to request medication refills and handle prescriptions efficiently. This functionality removes friction from the medication management process, ensuring patients can get their prescribed medications without delays or confusion.

5. *Leverage Data for Better Health Outcomes*

The system will incorporate predictive analytics, including machine learning (ML) models, to assess medication adherence risks and provide valuable insights for clinicians. This data-driven approach will assist healthcare providers identify patients at higher risk for non-adherence, enabling proactive interventions that can improve health outcomes.

By achieving these objectives, the HealthSmart Medication Management System will empower both patients and healthcare providers to work together more effectively, leading to improved adherence to prescribed medications, better health outcomes, and improved efficiency in a critical healthcare process.

3. Scope Management

The scope of the Healthsmart Medication Management System project has been defined to ensure that it focuses on its primary goal: improving medication adherence and supporting patients in managing their medications. To maintain this focus and avoid scope creep, these exclusions help ensure that the system remains with its objectives.

3.1. Exclusions

1. Communication Service: The system will not implement a communication service internally. While communication is a vital component of patient care, the message business object that facilitates communication is outside the scope of this project. The primary focus of the Healthsmart Medication Management System is on medication tracking and adherence, not on direct communication

between patients and healthcare providers. To address communication needs, the system will integrate with a third-party communication service that is preferably HIPPA compliant, secure, and reliable. This ensures that communication needs are met while maintaining the system's focus on its core functionality. Additionally, this service should be capable of integrating with Spring Boot and should align with the security realm of the Healthsmart Medication Management System, ensuring seamless integration while maintaining robust security standards.

2. Pharmacy Service: The Healthsmart Medication Management System will exclude the implementation of a pharmacy service. While pharmacy management, including medication dispensing and fulfillment, is a critical component of the healthcare system, it is considered out of scope for this project. The primary function of the system is to help patients track their medications and dosage schedules, rather than to directly handle the pharmacy-related processes. Although the system will interface with external pharmacy services for refill requests, it will not be responsible for the medication fulfillment process. Requests for medication refills will be routed through the RESTful API of the Refill Service, which external pharmacy applications can connect to. This ensures that pharmacy-related tasks remain within the scope of pharmacy systems, and not the HealthSmart Medication Management System.

3. Infrastructure: The HealthSmart Medication Management System will exclude the implementation of infrastructure within the services themselves. The project will not be responsible for building or managing the physical infrastructure needed to support the services. This will be done through Kubernetes clusters deployed on AWS Kubernetes Engine or a similar platform, allowing the system to focus on its functional requirements.

4. Patient Portal Features: The HealthSmart Medication Management System will not offer comprehensive patient portal features, such as appointment scheduling, a bill center, electronic health record (HER) viewer. The system's primary domain is medicine, not broader healthcare management (i.e., appointment, bill). Features like billing and appoint viewing will be handled by other specialized healthcare platforms, such as EHR systems or patient portal solutions, ensuring that the system focuses on its core medication management goals.

5. EHR Aggregation: The HealthSmart Medication Management System will not be responsible for managing EHR aggregation. While integrating patient health records is critical for a comprehensive healthcare solution, EHR aggregation will be handled by an InterSystems interface, ensuring consistent formatting of records. This system provides wider access to patient records while also offering fault tolerance, which is managed separately and remains outside the scope of

this project. The HealthSmart Medication Management System will interact with the aggregated data but will not manage the aggregation process itself.

Also, it is important to note that the patient records aggregated, in this case, will include medication records, allergy records, and other medication-related data, which are directly relevant to medication management. The system will rely on this aggregated data to provide accurate medication tracking, adherence monitoring, and dosage reminders, but the aggregation and formatting of these records will be handled by the InterSystems instance.

3.2. Core Focus

The HealthSmart Medication Management System will focus on enabling patients to manage their medications, track adherence, and receive timely dosage reminders. Additionally, the system will help clinicians manage patient medications, visualize adherence through various metrics, and process medication-related concerns quickly and efficiently.

While the system will integrate with third-party services for communication, external pharmacy services, and electronic health record (EHR) aggregation, its core functionality will remain centered on medication management. By excluding specific features outside of this core functionality, the system ensures it delivers a solution that meets its primary objectives.

4. Project Schedule/Deliverables

Following an Agile approach, the Healthsmart Medication Management System will implement features using two-week sprints. Each sprint will focus on specific features, with key deliverables at the end of the sprint cycle. Below is a high-level overview of the sprint schedule, key deliverables, and a brief description of each sprint:

Sprint Number	Sprint Length	Key Deliverables	Description
1	2 weeks	Project Management Plan, DevOps Pipeline Document, System Architecture and Design Document, Interface Specification, Software Requirement Specification	Sprint 1 will focus on preparing foundational documentation that will guide the development process. This sprint will ensure that all documentation is in place to facilitate smooth development in subsequent plans.

2	2 weeks	GitHub Workflows, Agent, and Jobs; CD Pipeline	Sprint 2 will focus on setting and delivering the CD pipeline using GitHub Workflows along with its respective components.
3	2 weeks	Infrastructure for cross-cutting concerns	Sprint 3 will focus on implementing key infrastructure components to solve cross-cutting issues like observability, monitoring, logging, and security.
4	2 weeks	Infrastructure for EHR aggregation, communication service and	Sprint 4 will focus on implementing infrastructure services that are not directly part of the

		pharmacy service integration	business logic but enable key features and services to function efficiently.
5	2 weeks	Medication Service	Sprint 5 will focus on implementing the medication service, which acts as the central service for the medication business object.
6	2 weeks	Medication Management Service	Sprint 6 will focus on implementing the medication management service, a backing service of the Medication Service. It provides operations for providers to make changes to

			prescriptions and adjust dosages. Also, it will enable patients to view their medication records.
7	2 weeks	Medication Scheduling Service	Sprint 7 will focus on implementing the medication scheduling service, a backing service of the Medication Service. It provides operations for clinicians to manage their patients' schedules. Also, it will enable patients to view their medication schedules.
8	2 weeks	Notification Service	Sprint 8 will focus on implementing the

			notification service, a backing service of the Medication Service. It instantiates a notification, a domain object, that handles the sending of alerts and reminders to users.
9	2 weeks	Notification Management Service	Sprint 9 will focus on implementing the notification management service, a backing service of the Notification Service. It provides operations for clinicians to configure, manage, and customize

			notifications for their patients. This service will focus on aspects like the type of reminders, notification timing, and other relevant configurations, ensuring clinicians can tailor patient communication efficiently.
10	2 weeks	Communication Service	Sprint 10 will focus on implementing the communication service, a backing service of the medication service. It is also associated with the notification

			service to alert patients when they have new messages from their physician and notify physicians when a patient has sent a new message. It provides operations, such as sending messages or a photo, for patients to communicate with their physicians, and vice versa.
11	2 weeks	EHR Aggregator Service	Sprint 11 will focus on implementing the EHR aggregator service, a backing service of the

			medication service. It is associated with the Notification Service to alert patients and physicians when new records are loaded into the system. The service will interact with external systems to collect, standardize, and integrate medication-related records.
12, 13	4 weeks	Angular SPA, responsive UI	Sprints 12 and 13 will focus on implementing the thin web-browser client that provides

			an interface for users to invoke functionality.
14, 15	4 weeks	Medication Non-Adherence Risk Service API, ML model, and Service	Sprints 14 and 15 will focus on implementing a ML model that asses the risk of medication non-adherence in patients. This model will provide clinicians with a risk score for each patient, indicating the likelihood that the patient will adhere to their prescribed medication schedule.

5. Resource Management

The HealthSmart Medication Management System project has a diverse set of needs, including highly specialized technical expertise, reliable infrastructure, and sufficient funding, all of which must be managed to ensure the project progresses as planned. Given the complexity of the system, careful resource allocation is essential to meeting milestones, adhering to sprint schedules, and staying within the allocated budget.

This section will analyze the following resource sections:

1. Human resources
2. Technological resources
3. Financial resources

5.1. Human resources

This project requires a specialized skill set to ensure the effective design, development, and deployment of the HealthSmart Medication Management System. Given the multi-disciplinary nature of the project, expertise is needed not only in software development, but also in healthcare systems. These diverse roles and responsibilities will be fulfilled by Joshua D'Agostino, an aspiring Java Web Application developer who is passionate about leveraging technology to address challenges in the healthcare sector.

D’Agostino’s skills and passion for software development and healthcare will be critical in delivering the project within the specified timeline. As the sole team member, D’Agostino will handle all aspects of the system’s lifecycle, including:

1. Software Development: Designing and developing the full-stack of the HealthSmart Medication Management System, including the Angular SPA, backend services, and infrastructure used for cross-cutting concerns. This involves ensuring the integration of services such as medication tracking, patient-provider communication, and external pharmacy services.
2. Healthcare Systems Expertise: D’Agostino will ensure that the system complies with essential healthcare industry standards such as HIPPA and FHIR (Fast Healthcare Interoperability Resources). This will ensure that the system properly handles sensitive health data while aligning the medication management features with existing clinical workflows.
3. Security: D’Agostino will implement critical security requirements, such as data encryption in transit, user authentication, and user authorization through role-based access control (RBAC), to protect patient data and ensure privacy. By employing secure development practices, such as regular vulnerability assessments and source code views, D’Agostino will work to eliminate potential security risks.
4. DevOps: As a DevOps practitioner, D’Agostino will manage the development, testing, and deployment process for the HealthSmart Medication Management System. This includes setting up cloud infrastructure, automating builds, and establishing CI/CD workflows. By leveraging DevOps practices, D’Agostino will ensure new features are

delivered promptly and the system remains scalable, reliable, and easily maintainable over time.

With these key responsibilities, D'Agostino will manage the development and execution of the project, ensuring it remains on schedule and delivers high quality results.

5.2. Technological resources

The HealthSmart Medication Management System is built on top of Spring Boot leveraging a service-oriented architecture (SOA) to facilitate features such as scalability, maintainability, and modularization. The architecture is designed to incorporate both imperative and reactive programming paradigms, allowing the system to handle service interactions efficiently while catering to different needs in terms of real-time responsiveness and robustness

As such, it uses a comprehensive array of frameworks and ecosystems to ensure scalability, reliability, and maintainability. These technologies facilitate the development, deployment, monitoring, and security of the system. Below is an overview of the key frameworks and tools integrated into the project:

1. Spring Boot

Spring Boot is the backbone of the HealthSmart Medication Management System. It simplifies the development of Java-based applications by providing production-ready configurations, embedded servers, and extensive libraries that accelerate the application's development cycle. Spring Boot ensures that services are modular and can scale independently, adhering to a service-oriented architecture (SOA).

2. Reactor4J Framework

Reactor4J enables reactive programming in the system, providing a non-blocking foundation for building highly responsive and scalable applications. This framework ensures that the system can handle concurrent requests with minimal latency, improving the overall performance and scalability of the HealthSmart Medication Management System.

3. Spring Boot Actuator

Spring Boot Actuator adds production-ready features, such as health checks, metrics, and application environment details. It helps monitor and manage the system, providing insights into the health of the application and enabling fast detection and resolution of any issues.

4. Spring Cloud Bus

Spring Cloud Bus is used for distributed messaging, enabling communication between microservices. It simplifies the propagation of state changes or events across services, especially in a cloud-native environment. It ensures that the services are synchronized and can handle changes in real time.

5. Docker

Docker enables containerization of the HealthSmart Medication Management System services, ensuring consistency across different environments. Docker containers provide an isolated environment for each service, which helps streamline deployment, scaling, and continuous integration/continuous deployment (CI/CD) processes.

6. Paketo Buildpacks

Paketo Buildpacks provide a standardized way to build and deploy Spring Boot applications on cloud platforms. It simplifies the process of containerizing Java applications, allowing for optimized runtime environments that ensure scalability and resilience.

7. Spring Boot Test with JUnit, Mockito, and Testcontainers

The combination of JUnit, Mockito, and Testcontainers is used for unit and integration testing. JUnit provides the test framework, Mockito handles mock objects, and Testcontainers facilitates the integration of Dockerized services for real-world testing scenarios, ensuring the system performs reliably before deployment.

8. Grafana Stack

Grafana is used for monitoring and visualizing metrics collected from the system. It integrates with **Prometheus** for real-time metrics collection and helps provide insights into system performance, resource usage, and service health, enabling proactive management and issue resolution.

9. GitHub (Actions, Workflows, Agent, Repositories)

GitHub serves as the source control and collaboration platform for the project. GitHub **Actions** and **Workflows** are used for automating CI/CD pipelines, testing, and deployments.

GitHub Repositories manage the project's code, while Agents facilitate the execution of tasks in the CI/CD pipeline.

10. Spring Cloud Stream

Spring Cloud Stream is used to simplify event-driven architecture in the system. It helps services communicate asynchronously, ensuring that events, such as patient medication updates or communication between patient and healthcare providers, are efficiently propagated across the system.

11. Spring Cloud Function

Spring Cloud Function enables the development of serverless functions that can be triggered by events or messages. It is used for creating lightweight microservices in the system, enabling flexibility and scalability in handling specific actions or data processing tasks.

12. RabbitMQ

RabbitMQ is the messaging broker used to facilitate communication between microservices in the system. It supports asynchronous message queues that ensure decoupling between services, improving fault tolerance, and enabling event-driven behavior.

13. Keycloak

Keycloak is the identity and access management solution used to handle user authentication and authorization. It provides support for OAuth2 and OpenID Connect, ensuring that the system enforces strong security standards and supports single sign-on (SSO) for both patients and healthcare providers.

14. Spring Security

Spring Security integrates with Keycloak to ensure that the HealthSmart Medication Management System is secure. It handles user authentication, authorization, and other security features like RBAC (Role-Based Access Control), ensuring that data privacy and security regulations, such as HIPAA, are met.

15. PostgreSQL - Databases

PostgreSQL serves as the relational database management system (RDBMS) for storing persistent data related to medications, patient profiles, appointments, and other critical information. PostgreSQL is highly reliable, supports complex queries, and provides robust transaction handling for the application.

16. Spring API Gateway

The Spring API Gateway acts as a single entry point for all requests to the system's services. It routes traffic to the appropriate service, handles security, rate-limiting, and can also serve as a proxy for client communications, helping centralize API management.

17. Redis

Redis is used as an in-memory data store for caching frequently accessed data and session management. It enhances performance by reducing database load and ensures faster response times for user requests.

18. Spring Cloud Circuit Breaker

Spring Cloud Circuit Breaker is employed to improve the resiliency of the HealthSmart Medication Management System. It helps prevent cascading failures in the event of service disruptions by gracefully handling service outages and providing fallback mechanisms.

19. Maven

Maven is used for managing the project's build lifecycle. It handles dependencies, compiles code, and generates artifacts that can be deployed in different environments, ensuring a consistent and reliable build process.

20. Resilience4J

Resilience4J is integrated with the system to provide additional fault tolerance. It includes features such as retries, timeouts, and circuit breakers to ensure the system remains resilient under high load or during intermittent failures.

21. Angular

Angular is the front-end framework for building the web browser-based user interface of the HealthSmart Medication Management System. It enables the development of a dynamic, single-page application (SPA) that interacts with back-end services through REST APIs.

22. Spring Reactive Stream

Spring Reactive Stream handles the asynchronous and non-blocking streams of data in the system. It enables the efficient processing and delivery of data to clients, ensuring that the system can handle high concurrency and large volumes of requests without performance degradation.

23. Spring Data R2DBC & Spring Data JDBC

Spring Data R2DBC is used to access relational databases reactively, while Spring Data JDBC simplifies the usage of JDBC for database access in a more traditional, blocking manner. Both frameworks ensure efficient database interaction depending on the service’s requirements.

24. Flyway

Flyway is used for database migrations. It ensures that the database schema is updated in a consistent, version-controlled manner as the system evolves.

25. Kubernetes

Kubernetes is used to orchestrate the deployment, scaling, and management of containerized services. It ensures that the HealthSmart Medication Management System can scale horizontally as demand increases, while managing the overall health of services in production.

26. OAuth2/OpenID Connect Frameworks using Spring Security

OAuth2 and OpenID Connect provide secure and standardized authentication mechanisms for the HealthSmart Medication Management System. These frameworks ensure that patient and provider data is securely protected, and sensitive information is accessible only to authorized users.

27. Prometheus

Prometheus is used to collect real-time metrics from the system, including service health, usage patterns, and system performance. These metrics can be visualized in Grafana to help monitor system behavior and ensure optimal performance.

28. Tempo

Tempo is integrated for distributed tracing, providing visibility into the flow of requests across microservices. This helps identify bottlenecks or latency issues, allowing for more effective debugging and optimization.

29. Spring Boot Micrometer

Micrometer provides instrumentation for Spring Boot applications, collecting and exporting performance metrics to monitoring systems like Prometheus, Grafana, and others. This allows the system to track important metrics like response times, error rates, and throughput.

30. FluentBit

FluentBit is a log forwarding and aggregation tool used to centralize log data from the system's various microservices. It forwards logs to a backend system such as Loki, enabling efficient log management and analysis.

31. Loki

Loki is a log aggregation system used alongside FluentBit. It collects, stores, and indexes logs, enabling centralized logging for easier troubleshooting and monitoring.

32. Kustomize

Kustomize is used for managing Kubernetes configurations in a more declarative and reusable manner. It enables the efficient customization of deployment resources, ensuring that environment-specific configurations are easily managed.

5.3. Financial Resources

The Healthsmart Medication Management System is a highly specialized project that requires allocation of financial resources to support its design, development, deployment, and maintenance. In the following sections, key financial resource areas will be scrutinized.

1. Development Costs

- Personnel Costs: Since the project is being handled by Mr.D'Agostino, costs will include his time for software development, testing, and deployment
- Tools & Software Licenses: Some development tools and frameworks may require licenses or subscriptions, including IDEs, testing tools, or cloud-based services
- Security Tools & Audits: Ensures that the system complies with security standards such as HIPPA and implements the necessary security protocols will require investment in security audits, penetration testing, and tools

2. Infrastructure Costs

- Cloud infrastructure: The system will require cloud resources for hosting services, databases, and containers. Providers will incur costs related to compute, storage, and networking services.
- Third party services: Integration with external services may require subscription fees or per-transaction costs
- Monitoring and Logging Tools: Services such as Grafana, Prometheus, and Loki will require cloud infrastructure to collect, store, and visualize system metrics. If using managed services, there may be monthly fees associated with the usage patterns in these platforms.
- DevOps & CI/CD Tools: Financial resources will also be allocated for services like GitHub Actions, Docker, and the associated workflows that automate builds, deployments, and tests

3. Operational Costs

- Maintenance: post-launch maintenance will include system updates, bug fixes, and modifications based on user feedback.
- Support and training: continuous user support for clinicians and patients, including creating user manuals and handling support requests

4. Compliance and Regulatory Costs

- Healthcare Compliance: compliance with healthcare regulations such as HIPPA, FHIR, and data protection laws will involve auditing, documentation, and periodic reviews to ensure compliance with the standards.

6. Quality Management

The HealthSmart Medication Management System will be designed to meet both user expectations and industry standards, ensuring the delivery of a high quality, user-centric platform. The core goal is to create a system that serves the needs of all user groups, including patients, healthcare providers, and system administrators, while ensuring it adheres to the best practices and regulations in the healthcare and software development industries. In the section below are key components to ensure the system meets its quality objectives:

1. User-Centered Design:

- Objective: To ensure the system meets the needs of its diverse user groups, including patients, healthcare professionals, and administrators
- Approach: To achieve this objective, our team will undertake a thorough process of user research, which involves conducting interviews and surveys with real users from each user group. This research aims to capture the preferences, needs, and pain points of each user group.

  Once we have gathered sufficient data, we will create functional requirements in the form of user stories that represent the journeys of these users within the system.

These stories will serve as a foundational tool in identifying the features and functionalities that need to be deployed.

2. Code Quality

Objective: To ensure that the system's code is clean, readable, and maintainable and adheres to both coding standards and industry best practices

Approach: To ensure the cleanliness and maintainability of the system's codebase, developers will integrate extensive code analysis into the deployment pipeline. This automated process will continuously monitor the quality of the code, identifying issues related to code structure, performance, security vulnerabilities, and potential bugs early in the development cycle. If any code does not meet the predefined quality criteria, the system will prevent the commit from passing and trigger a failure in the pipeline, prompting developers to address and resolve issues before committing.

To ensure the readability of the system's codebase, developers will thoroughly document critical system operations and components. By adopting a modular approach to development, each section of the code will be self-contained, allowing others to comprehend individual parts without needing to understand the entire system.

3. Scalability and Robustness

Objective: To ensure that the system is robust and can scale based on demand

Approach: The HealthSmart Medication Management System will be built with a cloud-native architecture to ensure scalability, flexibility, and efficient resource management. Leveraging Kubernetes for container orchestration will allow the system to efficiently manage and scale its services based on demand. Kubernetes provides automatic scaling, load balancing, and self-healing mechanisms, ensuring the system remains highly available.

Since the system's state is managed by the databases of individual services, each service can be independently scaled depending on the load and requirements, ensuring that resources are utilized optimally.

By decoupling the state from the services themselves and utilizing Kubernetes, the HealthSmart Medication Management System can take advantage of cloud elasticity, scaling up or down as needed to meet the varying traffic demands without affecting performance.

4. Automated Testing and Continuous Integration

The HealthSmart Medication Management System employs a comprehensive testing strategy that spans unit tests, integration tests, security tests, and other specialized tests to ensure that every component of the system is validated throughout the development lifecycle. These tests are systematically integrated into the CI/CD pipeline, with each service having its own tailed suite of tests designed to verify functionality, performance, and security.

For example, the pipeline for the API of the Medication Suite would include a test suite to validate each operation. The suite would check that RBAC is properly enforced for each endpoint, ensuring resources are made available to users based on their roles. It would also cover standard API checks such as exception handling and data validation.

Automating the testing process within the pipeline provides developers with rapid feedback, significantly reducing the time between code changes and identification of issues.

5. Security

The HealthSmart Medication Management System integrates Keycloak OAuth2/OpenID Connect for user authentication and authorization, ensuring that access control is robust and secure. Using RBAC, the system guarantees that users can only access resources relevant to their role, preventing unauthorized access to sensitive information.

For instance, patients are granted access only to their own health information, such as medication records, schedules, and reminders, while clinicians can access patient data that is relevant to their practice, but only for the patients they manage.

By implementing OAuth2/OpenID Connect with Keycloak, the system ensures that authentication is handled securely, with tokens issued to verify identity. Coupled with RBAC, this setup ensures fine-grained access control, meaning users will only be able to perform actions or view data their role permits.