Vrddhopasthaan: Senior Citizen Health & Reminder App

Software Architecture Document
Version 1.0
Revision History

Date	Version	Description	Authors
22/March/2025	1.0	Software Architecture Document generated using Rational SoDA template and Rational Rose model.	Avilasha Goswami, Deep Habiswashi, Kaushiki Sarakar, Soumyadeep Dutta

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

The purpose here is to present the architectural blueprint for the Vrddhopasthaan app. It explains how the system will support health management for seniors by offering functionalities such as medicine reminders, emergency SOS, health record tracking, appointment scheduling, and caregiver monitoring.
This section outlines the architectural vision that underpins the design and development of an accessible, secure, and responsive application for senior citizens

1.2 Scope

The scope covers a multi-platform mobile and web app that serves as a health reminder and management assistant.It includes integration with external medical databases and emergency services while ensuring usability for seniors (aged 60+).The project will provide features like personalized medication alerts, secure health record storage, appointment scheduling, and a dedicated emergency SOS function

1.3 Definitions, Acronyms and Abbreviations

Definition of key terms such as:

- **UI:** User Interface components tailored for senior users.
- **API:** Application Programming Interfaces used to integrate with third-party health services.
- **2FA:** Two-Factor Authentication for secure login.
- **HealthKit/VoiceKit:** Native frameworks used for health data and voice command features.

This ensures that all stakeholders have a common understanding of technical and domain-specific terminology.

1.4 References

1. Applicable references are:

Health Data Security Standards:

- **GDPR Overview:**
  GDPR.eu
  Offers an accessible overview of the General Data Protection Regulation, which is crucial for any app handling sensitive data.

iOS Framework Documentation:

- **HealthKit:**
  Apple Developer - HealthKit Documentation
  Details on integrating health data management within your app.

- **UserNotifications:**
  Apple Developer - UserNotifications Framework
  Essential for implementing local notifications, such as medicine reminders.

- **SiriKit:**
  Apple Developer - SiriKit
  Useful for incorporating voice command features into your application.

Research and Guidelines on Accessible Mobile Applications for Older Adults:

- **Mobile Health Applications for Older Adults: A Systematic Review of Interface and Persuasive Feature Design:**

  Read the JAMIA Article

- **Prevalence and Correlates of Medication Reminder App 'Use and Use Intention' among Older Adults:**
  View the Study on ScienceDirect
  Explores factors influencing the adoption of reminder apps by seniors.

- **Co-Designing a Medication Notification Application with Multi-Channel Reminders:**
  Access the preprint on arXiv

- **WHO Global Report on Ageing and Health:**
  WHO Report 2015

2. Architectural Representation

The architecture for Vrddhopasthaan will be depicted using several complementary views.**Use-Case View:** Demonstrates key user interactions such as registration, setting reminders, activating emergency SOS, etc. **Logical View:** Outlines the structure of the app, including UI layers, business services (e.g., reminder scheduling, notification handling), and data persistence.**Process View:** Shows the runtime behavior, like concurrent tasks for notifications, voice command processing, and health record synchronization.**Deployment View:** Maps the application's components to physical nodes such as iOS devices, cloud servers, and API gateways.

3. Architectural Goals and Constraints

- **Goals:**
  Accessibility: Ensure that the UI is senior-friendly (large fonts, high contrast, voice guidance).
  Reliability & Performance: Deliver notifications within 1 second and maintain high uptime (targeting 99.9%).
  Security: Use encrypted storage and secure authentication (including 2FA or Sign In with Apple).
  Integration: Seamlessly connect with third-party health databases and emergency services.

- **Constraints:**

  Platform Limitations: Must work efficiently on low-power devices (like older iOS devices).

Connectivity: Requires reliable internet for cloud data sync and emergency communications.
Compliance: Must adhere to healthcare privacy laws and data security standards.

4. Use-Case View

A description of the use-case view of the software architecture. The Use Case View is an important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

The Vrddhopasthaan use cases are:
- User Registration & Login
- Medicine Reminders
- Emergency SOS Activation
- Health Record Tracking
- Appointment Scheduling
- Caregiver Mode

These use cases are initiated by the patients and doctors as per their use cases.

4.1 Architecturally-Significant Use Cases

4.1 Architecturally-Significant Use Cases

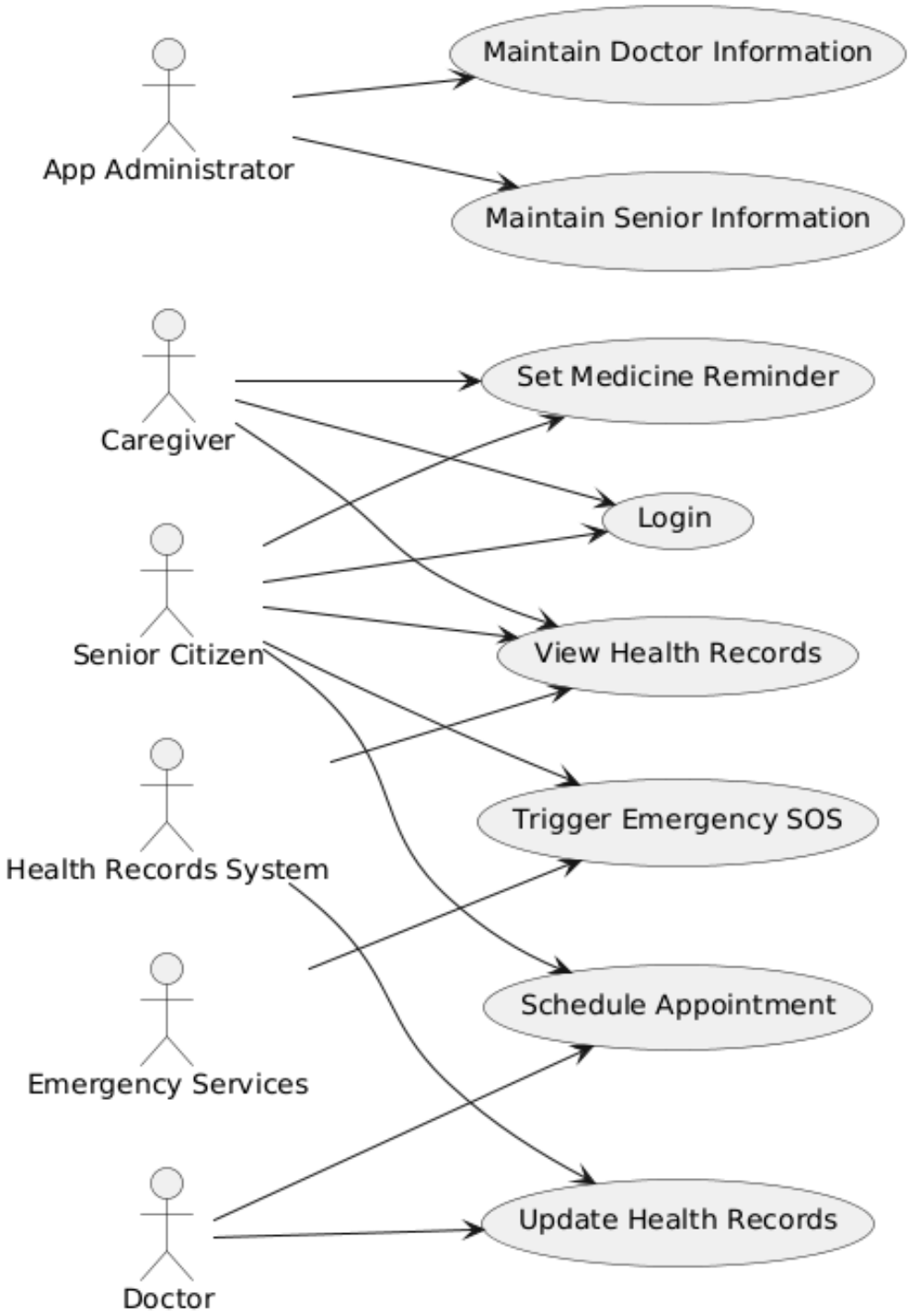


Diagram Name: Architecturally Significant Use-Cases

4.1.1 Emergency SOS Activation
Brief Description:This use case allows a senior citizen to immediately trigger an emergency alert when needed. When activated, the app sends a notification to pre-configured emergency contacts and, if integrated, to local emergency services.

Actors: Senior Citizen (primary), Emergency Contacts, Emergency Service Integration (secondary)

Key Points: One-tap activation for urgent assistance.
Automatic inclusion of the user's current location (via GPS).
Immediate push notifications and/or SMS to contacts.

4.1.2 User Login
Brief Description:This use case describes how users (senior citizens, caregivers, or healthcare professionals) securely log into the Vrddhopasthaan app.

Actors: Senior Citizen, Caregiver, Health Professional

Key Points: Secure authentication with options for 2FA or biometric login (Face ID/Touch ID).User-friendly interface designed for ease of use by seniors.

4.1.3 Maintain User Profile & Health Information
Brief Description:This use case allows users to add, modify, or delete their personal and health-related information. This includes updating medication details, health metrics, and emergency contacts.

Actors: Senior Citizen (primary), Caregiver (as an agent)

Key Points: Secure management of sensitive health records.Support for multiple data types (e.g., text, numerical health metrics).

4.1.4 Set Medication Reminders
Brief Description:This use case enables a user (or their caregiver) to schedule personalized medication reminders. Reminders are set based on specific dosages and timing requirements.

Actors: Senior Citizen (primary), Caregiver (optional)

Key Points: Custom scheduling and repeat options. Integration with local notification systems to ensure timely alerts.Ability to edit or cancel reminders as needed.

4.1.5 Appointment Scheduling

Brief Description:This use case allows users to schedule, modify, or cancel appointments with healthcare providers. The system integrates with calendar services to send timely reminders.

Actors: Senior Citizen, Caregiver

Key Points: Easy-to-use calendar interface tailored for seniors.
Automated reminders and integration with external calendar APIs (e.g., EventKit). Support for modifications within designated add/cancel periods.

4.1.6 View Health Records/Progress
This use case enables a senior citizen to review their health records, including previous appointments, medication logs, and recorded health metrics.

Actors: Senior Citizen

Key Points: Clear, accessible presentation of historical health data.Options for detailed views and summary dashboards.Secure display with privacy controls.

4.1.7 Log Health Data / Update Health Metrics

Brief Description: This use case allows users or caregivers to input new health metrics—such as blood pressure, glucose levels, or weight—following medical check-ups or daily monitoring.

Actors: Senior Citizen, Caregiver

Key Points: Intuitive data entry forms designed for ease of use.Data validation and storage in secure local/cloud databases.Integration with HealthKit for automated syncing when available.

4.1.8 Manage Emergency Contacts & Caregiver Information

Brief Description: This use case allows users to maintain and update their list of emergency contacts and caregiver details.

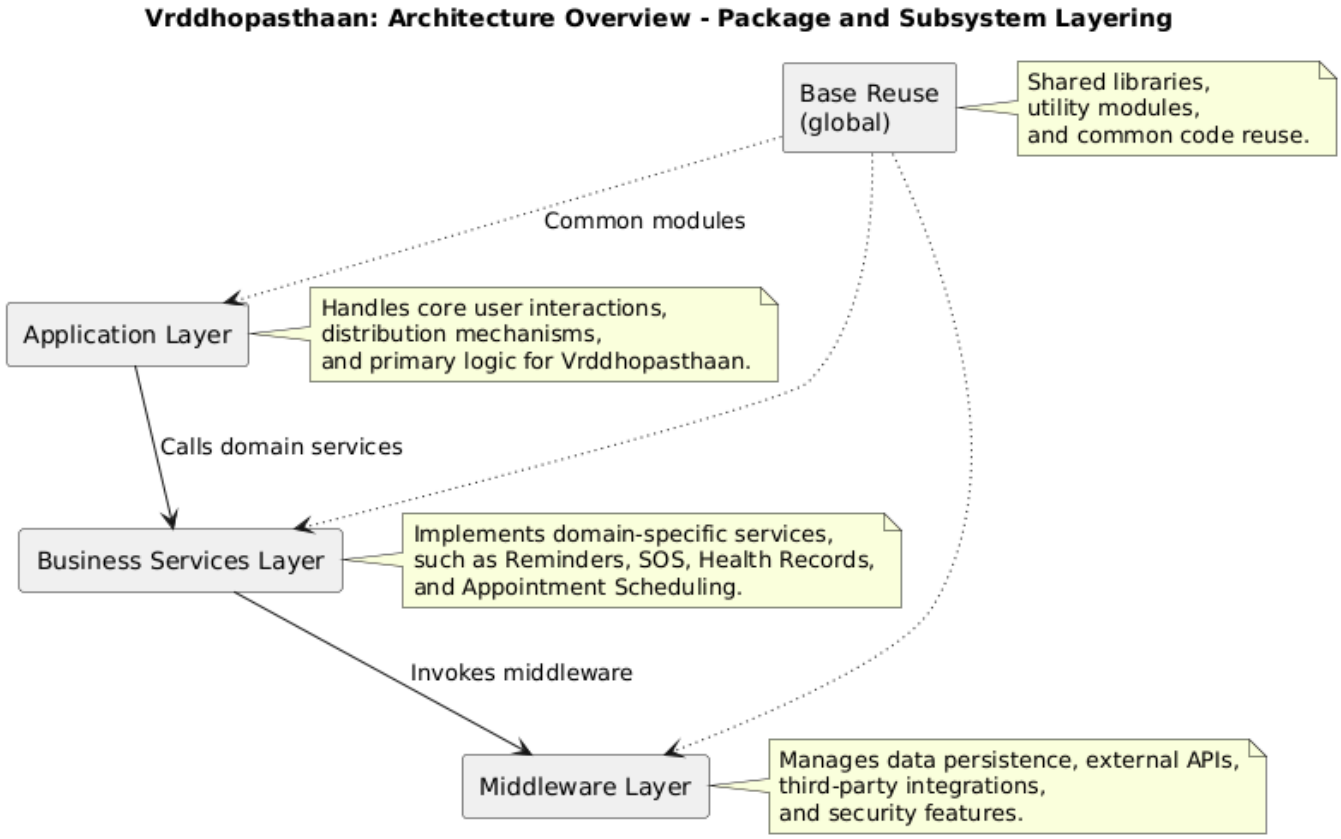
Actors: Senior Citizen (primary), Caregiver

Key Points: Add, modify, or delete contact information.Ensure that emergency contacts are readily accessible in case of an SOS activation.Privacy and security measures to protect sensitive contact data.

5. Logical View

A description of the logical view of the architecture. Describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.The logical view of the course registration system consists of the 3 main packages: User Interface, Business Services, and Business Objects.
The **User Interface Package** comprises all the components that users interact with. It focuses on simplicity and accessibility for senior citizens. The **Business Services Package** contains the core logic and controllers that bridge the user interface with the underlying data and external services. The **Business Objects Package** defines the data models and entities that represent the core concepts of the application.

5.1 Architecture Overview – Package and Subsystem Layering



5.1.1 Application Layer This layer encompasses all user-facing components, providing interfaces that facilitate user interaction with the system. It includes:

- **User Interface Components:** Screens and dialogs that allow users to perform tasks such as setting medication reminders, viewing health records, and initiating emergency alerts.

- **Accessibility Features:** Tools and settings designed to enhance usability for senior citizens, ensuring the application is intuitive and easy to navigate.

This layer depends on the Business Services layer to process user inputs and retrieve necessary data.

5.1.2 Business Services Layer Serving as the core of the application, this layer contains the business logic and controllers that manage the application's behavior. It includes:

- **Use Case Controllers:** Manage specific application behaviors such as scheduling reminders, processing health data, and handling emergency protocols.

- **Integration Services:** Handle communication with external systems like health data repositories, notification services, and emergency contact systems.

This layer interfaces with both the Application Layer (handling user interactions) and the Middleware Layer (managing data storage and retrieval).

5.1.3 Middleware Layer This layer provides the infrastructure that supports data management and communication between the Business Services layer and data storage systems. It includes:PMC

- **Database Access Components:** Manage connections to relational and object-oriented databases that store user information, health records, and application settings.

- **API Management:** Facilitates secure and efficient communication with external services, such as third-party health monitoring devices or cloud-based health data platforms.

The Middleware Layer ensures that data transactions are handled securely and efficiently, supporting the application's performance and reliability.

5.1.4 Base Reuse Layer This foundational layer includes utility classes and common functions that are reused across the application. It encompasses:BioMed Central

- **Utility Functions:** Common operations such as data validation, formatting, and logging that are utilized by various components of the application.

- **Design Patterns:** Standardized solutions to common problems, ensuring consistency and maintainability in the application's codebase.

6. Process View

A description of the process view of the architecture. Describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.The Process Model illustrates the

course registration classes organized as executable processes. Processes exist to support student registration, professor functions, registration closing, and access to the external Billing System and Course Catalog System.

6.1 Processes

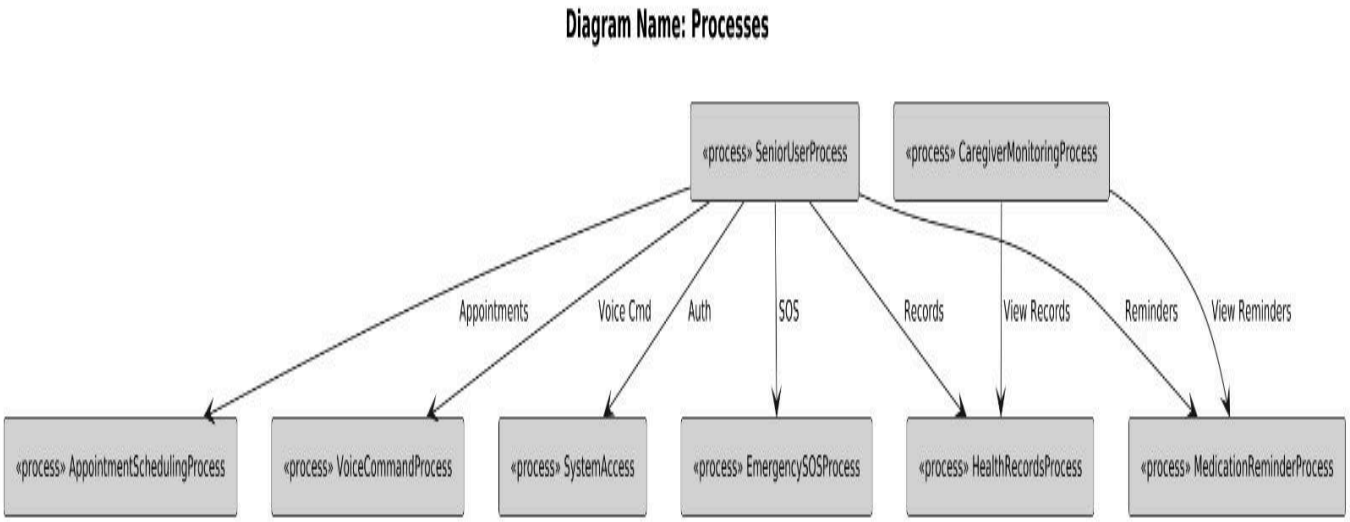


Diagram Name: Processes

6.1.1 MedicationReminderProcess

This process manages the scheduling, caching, and asynchronous dispatch of medication reminders.

- Details:
    - Retrieves and caches medication data to improve performance.
    - Utilizes separate threads (e.g., ReminderCache) to asynchronously fetch and schedule notifications.

- Analysis Mechanisms:
    - Notification scheduling and local caching to ensure reminders are delivered on time.
- Requirements Traceability:
    - The system shall deliver medication alerts within one second of the scheduled time.

6.1.2 EmergencySOSProcess

This process handles the immediate activation and processing of emergency alerts.

- Details:
    - When a user activates the SOS feature, this process promptly gathers the current location and dispatches notifications to pre-defined emergency contacts and, if available, directly to emergency services.
    - Functions as an adapter to external emergency communication systems.
- Analysis Mechanisms:
    - High-priority thread execution to ensure rapid response.
- Requirements Traceability:
    - The system shall ensure emergency alerts are processed and transmitted with minimal latency.

6.1.3 User Authentication Process

This process manages secure user login, registration, and session management.

- Details:
    - Authenticates users using standard protocols and supports advanced methods like biometric authentication (Face ID/Touch ID) or two-factor authentication.

    - Each user session is individually maintained and validated.

- Analysis Mechanisms:
    - Secure authentication and token validation integrated with iOS security frameworks.

- Requirements Traceability:
    - The system shall enforce strict authentication and maintain secure user sessions.

6.1.4 HealthRecordSyncProcess

This process synchronizes health data between the local device and cloud storage.

- Details:
    - Supports offline caching of health records and periodic updates to ensure data consistency.
    - Integrates with native frameworks such as HealthKit for seamless data handling.

- Analysis Mechanisms:
    - Asynchronous data synchronization and error-handling to guarantee data integrity.

- Requirements Traceability:
    - The system shall securely maintain and synchronize user health records with minimal data loss.

6.1.5 Appointment Scheduling Process

This process handles the scheduling, modification, and cancellation of healthcare appointments.

- Details:

    - Integrates with local calendar services (via EventKit) to manage appointment data.
    - Sends timely notifications to remind users of upcoming appointments.
- Analysis Mechanisms:
    - Calendar integration and notification scheduling for real-time updates.
- Requirements Traceability:
    - The system shall allow users to effectively schedule and modify appointments with accurate reminders.

6.1.6 Caregiver Monitoring Process

This process provides a dedicated channel for caregivers to monitor the health status and reminders of senior users.

- Details:
    - Offers a read-only interface displaying current medication schedules, health records, and emergency statuses.
    - Ensures that sensitive data is securely transmitted and accessible only to authorized caregivers.
- Analysis Mechanisms:
    - Real-time data push and secure access protocols.
- Requirements Traceability:
    - The system shall support remote monitoring capabilities for designated caregivers while ensuring data privacy.

6.2 Process to Design Elements

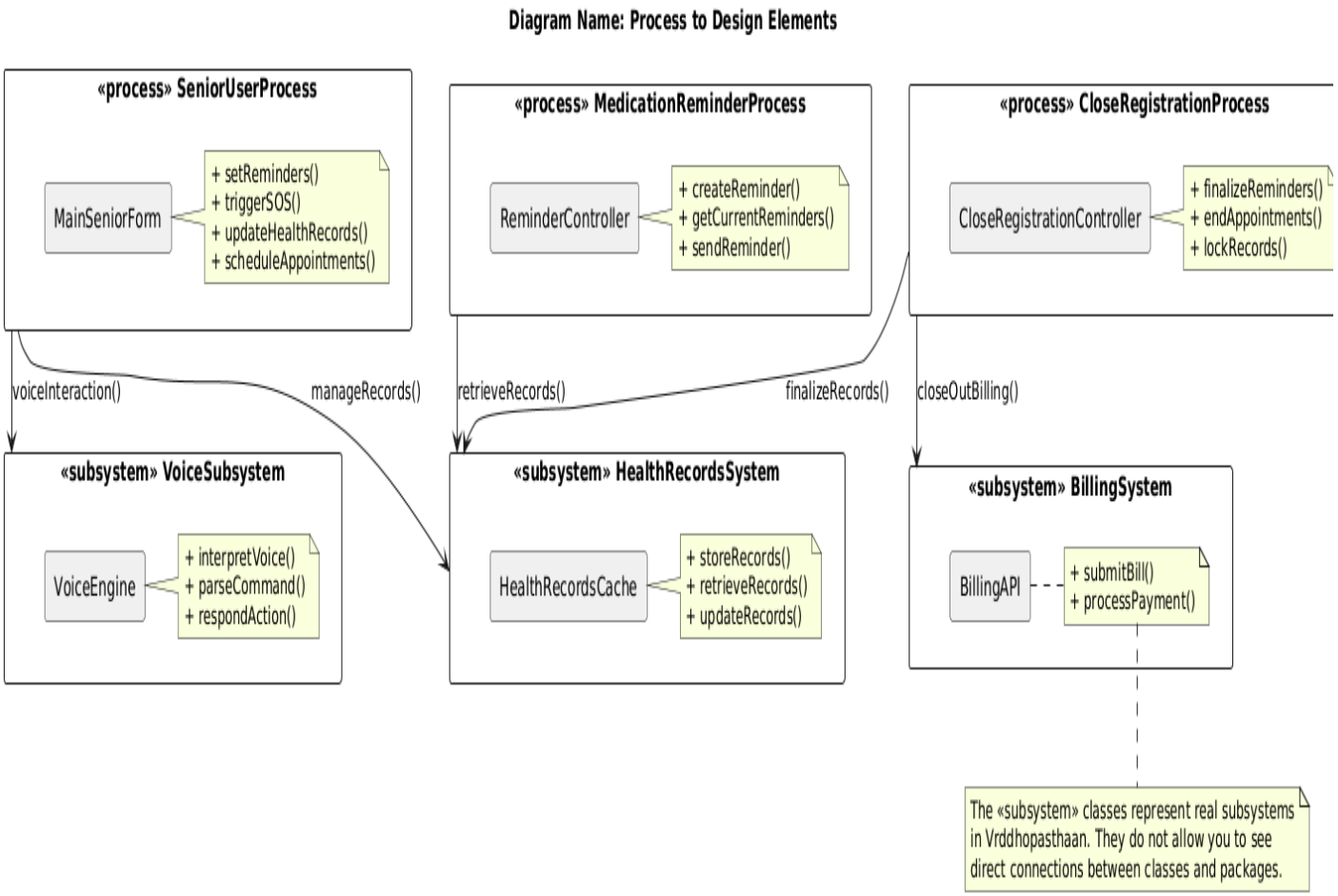


Diagram Name: Process to Design Elements

6.2.1 MedicationCache

- *Purpose:*
  Asynchronously retrieves and caches medication details and schedules from external health data sources.

- *Analysis Mechanisms:*

  - Persistency

  - External Health Data Integration

6.2.2 Reminder Cache

- *Purpose:*
  Asynchronously updates and retrieves scheduled medication reminders to ensure timely notifications.

- *Analysis Mechanisms:*

    - Persistency

    - Real-Time Notification Scheduling

6.2.3 Medication

- *Purpose:*
  Represents a medication entity, including details such as name, dosage, and frequency.

- *Analysis Mechanisms:*

    - Persistency

    - Integration with Health Data APIs

6.2.4 Medication Reminder

- *Purpose:*
  Defines a specific reminder for taking a medication, including timing and recurrence information.

- *Analysis Mechanisms:*

    - Persistency

    - Local Notification Scheduling

6.3 Process Model to Design Model Dependencies

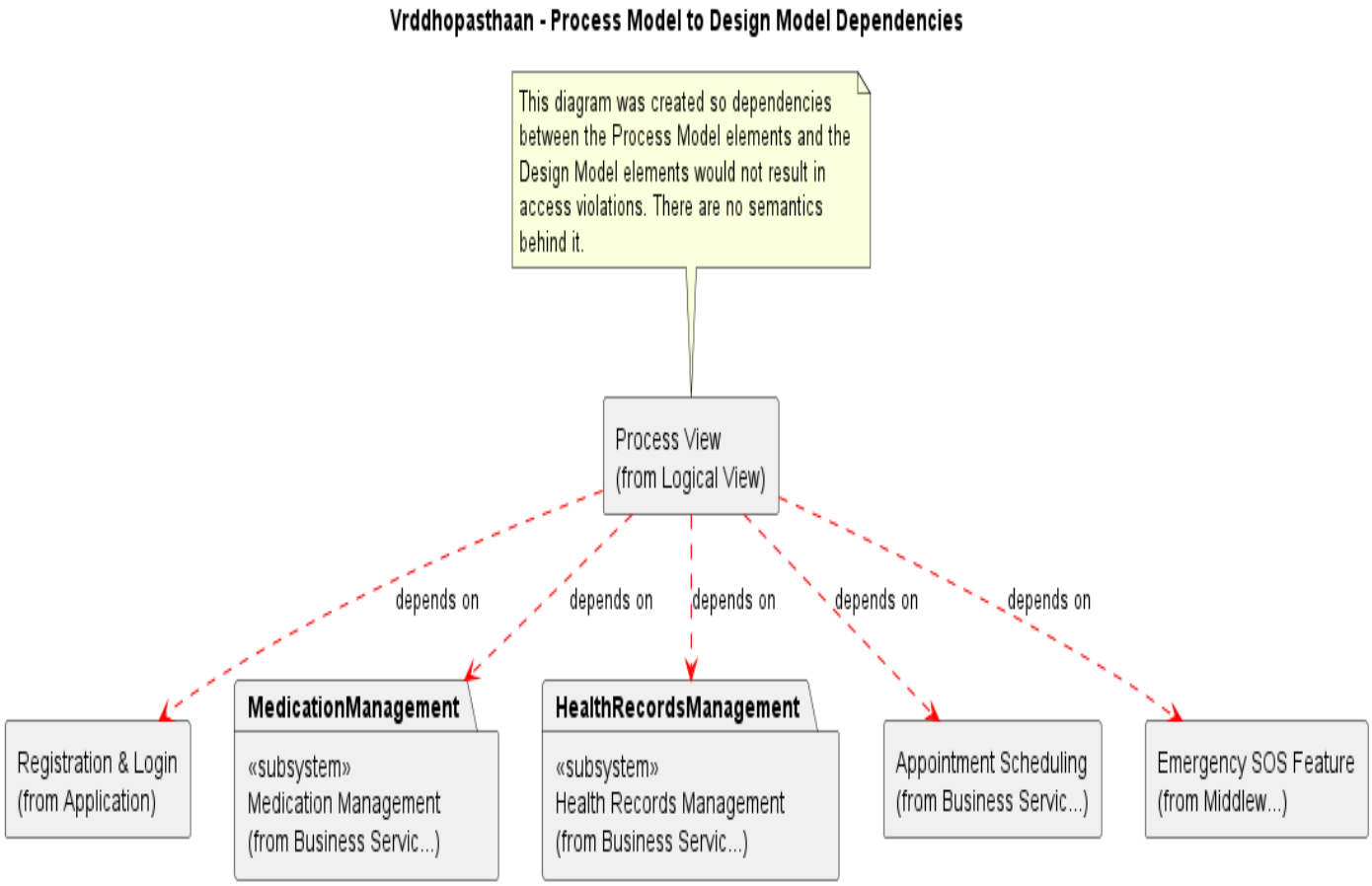


Diagram Name: Process Model to Design Model Dependencies

6.4 Processes to the Implementation

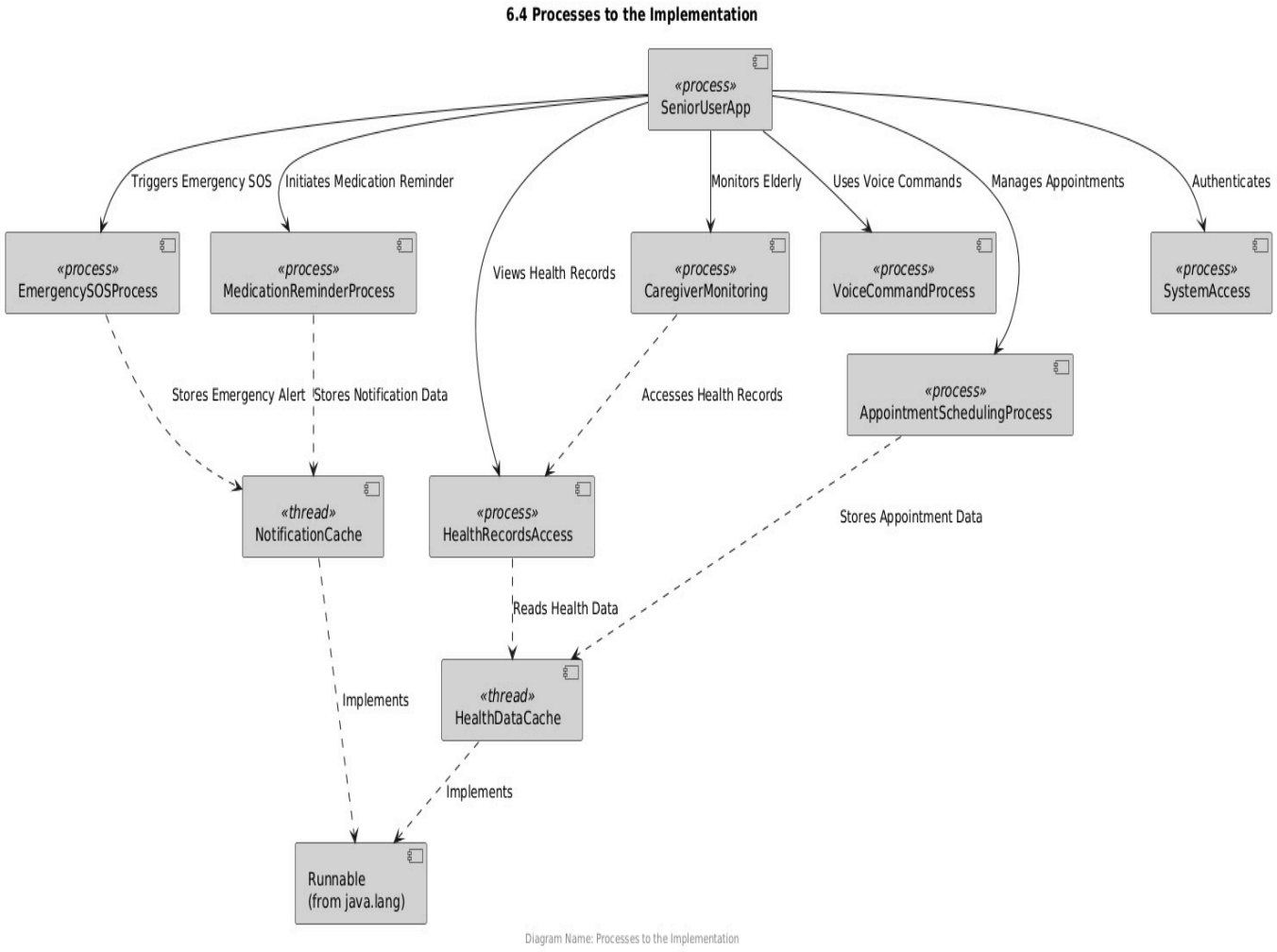


Diagram Name: Processes to the Implementation

6.4.1 Remote Abstraction:

*Defines a protocol for objects accessed over the network (e.g., API clients for health data).

*Only exposes specific methods for remote interactions.

*Supports multiple remote protocols to integrate external services.
6.4.2 Runnable Abstraction:
*Establishes a standard for tasks meant to run asynchronously (e.g., notification scheduling, data sync).
*Requires implementation of a `run()` method to start the task.

*Ensures consistent execution of background operations.
6.4.3 Thread Abstraction:
*Represents an execution unit that runs concurrently (e.g., using Grand Central Dispatch or NSOperation in iOS).
*Allows setting priority levels for tasks to ensure time-critical actions (like emergency alerts) are prioritized.
*Supports background processing, similar to daemon threads for ongoing operations.

7. Deployment View

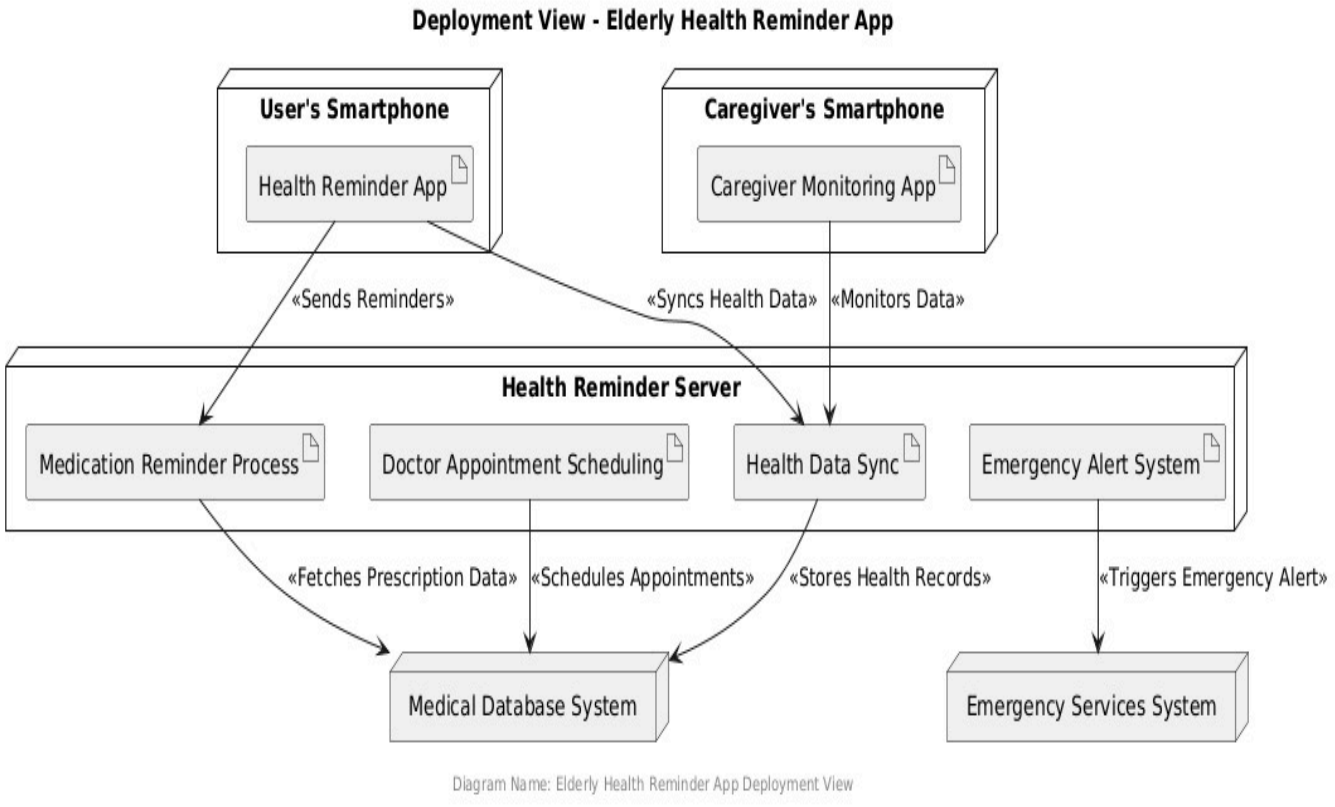


Diagram Name: Deployment View 7.1 of the APP

7.1 Mobile Device (iOS Client)

- Role:
    - Hosts the Vrddhopasthaan application for senior citizens.
- Components:
    - Application Layer: User interface screens for medication reminders, emergency SOS, health records, and appointment scheduling.
    - Local Business Services: Handles local tasks such as notification scheduling (using UserNotifications) and voice command processing (via SiriKit).
- Connectivity:
    - Communicates with the Cloud Backend for data sync and remote notifications.

7.2 Caregiver Web/Desktop Client

- Role:
    - Provides a companion interface for caregivers to monitor health records, reminders, and emergency alerts.
- Components:
    - Web-based or desktop interface that accesses read-only views of user data.
- Connectivity:
    - Connects to the Cloud Backend over the internet, ensuring secure data retrieval.

7.3 Cloud Backend Server

- Role:
    - Acts as the central hub for business logic, data management, and external integrations.
- Components:

    - Business Services Layer: Manages user authentication, appointment scheduling, health record synchronization, and emergency alert processing.

    - Data Persistence: Manages secure storage of user profiles, health records, medication schedules, and emergency contact information.

- Connectivity:

    - Interfaces with both Mobile Devices and Caregiver Clients.

    - Integrates with external APIs, such as Health Data Providers and Emergency Services Gateways.

7.4 External Health Data Providers

- Role:

    - Serve as external nodes that supply health metrics, medication data, or additional health insights.

- Components:

    - Legacy health databases or third-party APIs.

- Connectivity:

    - The Cloud Backend communicates with these external systems to enrich local data and support informed decision-making.

7.5 Emergency Services Gateway

- Role:

    - Processes and dispatches emergency notifications.

- Components:

    - A dedicated service (or API integration) that connects with local emergency contacts and services.

- Connectivity:
    - Directly triggered by the Emergency SOS process in the Mobile Device, then managed via the Cloud Backend for further processing and confirmation.

8. Size and Performance

- Resource Efficiency:
    - Optimized for low memory and disk usage on mobile devices (target app size <20 MB, low RAM footprint).
- Timely Notifications:
    - Medication and emergency alerts delivered within 1 second of their scheduled time.
- Scalability:
    - Designed to support a large number of concurrent users (e.g., up to 1000 simultaneous mobile users) with high availability.
- Network Efficiency:
    - Efficient data synchronization with cloud backend ensuring minimal latency during peak usage.

9. Quality

- Usability:
    - User interface is tailored for senior citizens, featuring large fonts, high contrast, and simple navigation.

    - Integrated online help and step-by-step guidance for all critical features.

- Reliability:
    - System uptime target of 99.9% with robust error handling and fallback mechanisms.
    - Mean Time Between Failures (MTBF) exceeds 300 hours.
- Security and Compliance:
    - Encrypted data storage and secure authentication (including 2FA/biometrics).
    - Adherence to healthcare data standards and privacy regulations (e.g., HIPAA, GDPR).
- Maintainability:
    - Modular architecture allows for easy updates and downloadable client-side upgrades over the internet.