Software Design Specifications

for

Oldie2GenZ

Prepared by:

Group Name: Fusion Force
Anjani Komaravelli
Chaitanya Deepthi
Shreeya Konda
Nikita Reddy
Rishita Chava
Akshay Palutla
Karthik
Pataneni

Document Information

Title:	**Website for Senior Citizen**		
Project Manager:		**Document Version No:**	
		Document Version Date:	
Prepared By:		**Preparation Date:**	**7th April 2025**

Version History

Ver. No.	Ver. Date	Revised By	Description	Filename

Table of Contents

1 Introduction

*This Software Design Specification (SDS) document outlines the design architecture and internal structure of the **Oldie2GenZ** web application. It is meant to guide developers, testers, and project stakeholders in understanding how the software is organized, how its components interact, and how each part fulfils specific requirements from the SRS.*

The document follows industry standards and provides design decisions, class structures, module interactions, data flow, and system constraints. It ensures that the software meets the expectations of usability, maintainability, security, and performance, especially tailored for senior citizens.

The design is based on the COMET methodology and uses UML for clear, structured representation. This document supports communication among team members and serves as a blueprint during development and future enhancements.

1.1 Purpose

*The purpose of this document is to define and describe the design of the **Oldie2GenZ** application. It explains how the system will be built to meet the functional and non-functional requirements outlined in the Software Requirements Specification (SRS).*

It serves several audiences:

- ***Developers**: For understanding how each module should be coded.*
- ***Testers**: To verify that each component functions as designed.*
- ***Project Managers**: To track design progress and ensure quality.*
- ***Reviewers/Professors**: To assess the completeness and clarity of the design.*

The structure of the document includes: use case views, logical and data views, exception handling, parameters, and quality considerations. It ensures a smooth transition from design to implementation.

1.2 Scope

*This SDS document applies to the full design of the **Oldie2GenZ** platform. It affects:*

- *Frontend and Backend design of the system*
- *Database schema and integration*
- *Monolithic architecture*
- *Notification, reminders, tutorial systems*
- *User interface elements tailored for senior citizens*

The document influences:

- *The way code is organized and structured*
- *User experience through large UI components*
- *How the system handles performance, accessibility, and scalability*

1.3 Definitions, Acronyms, and Abbreviations

- ***Oldie2GenZ**: The web application created to help senior citizens access online services.*
- ***SRS**: Software Requirements Specification – outlines what the software should do.*
- ***SDS**: Software Design Specification – describes how the software will be built.*

- ***COMET**: A methodology for modeling object-oriented software systems.*
- ***UML**: Unified Modeling Language – a standard way to visualize the design of a system.*
- ***DBMS**: Database Management System – software for storing and managing data (like PostgreSQL).*
- ***WCAG**: Web Content Accessibility Guidelines – standards for making websites accessible, especially for users with disabilities.*

1.4 References

This design document refers to the following materials:

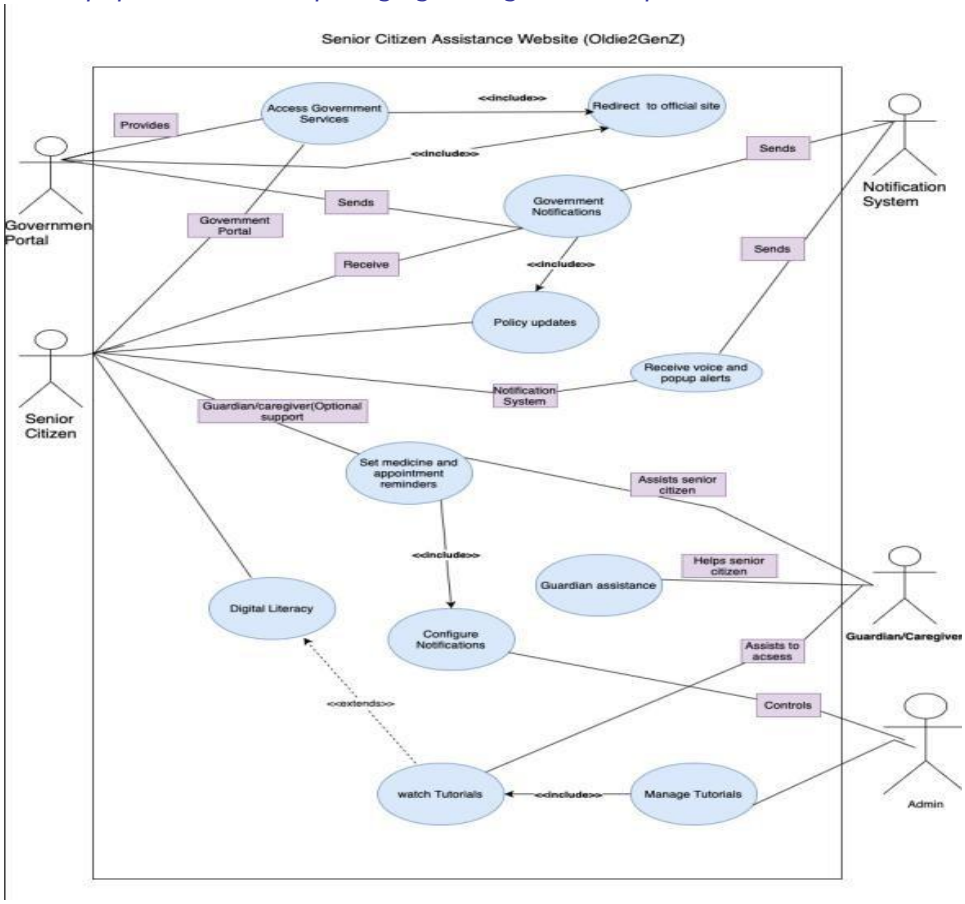
- ***IEEE Software Engineering Standards** – for structure and formatting.*
- ***Oldie2GenZ SRS Document** – for requirements and use cases.*
- ***Use Case and Class Diagrams** – for visual design references.*
- ***Monolithic Architecture Document** – for module breakdown.*
- ***COMET methodology** – for object modeling and design structure.*
- ***Government of India Portals** – such as Jeevan Pramaan and pension portals.*
- ***WCAG Guidelines** – to ensure senior-friendly and accessible UI.*

2 Use Case View

*The **Oldie2GenZ** platform is designed around several core functionalities essential to senior citizens. The use cases identified here are derived from the functional requirements and provide a basis for understanding how users will interact with the system.*

These use cases represent major features of the application and cover significant components in the system's architecture. Each use case involves interaction between the user (either the senior citizen or caregiver) and the various modules like the Reminder System, Notification System, Government Service Manager, and Learning Modules.

These use cases stress accessibility, ease of navigation, and personalized services for elderly users. They ensure the application fulfills its mission of bridging the digital divide for senior citizens.



2.1 Use Case

The key use cases for the Oldie2GenZ platform:

1. User Registration and Login
***Description**: Allows a new user (senior or caregiver) to register and log in securely.*

- *Steps:*
  1. *User opens the app and selects SignUp/login.*
  2. *Enters basic information.*
  3. *Verifies identity via password.*
  4. *Gets redirected to dashboard upon success.*

2. Set Medicine and Appointment Reminders
***Description**: Users (or caregivers) can set reminders for medication and appointments.*

- *Steps:*
  1. *Navigate to reminder section.*
  2. *Select type (medicine or appointment).*
  3. *Enter name,time,dosage etc.*
  4. *Save – system schedules notification.*
  5. *Reminder pops up with text alert at the scheduled time.*

***3. Access Government Services Description**:*
Redirects users to appropriate government portals like Jeevan Pramaan.

- *Steps:*
  1. *User selects the government services option.*
  2. *Browses categories (pension, health, etc.).*
  3. *Clicks on a service and gets redirected to the official site.*
  4. *Receives guidance/tutorial if needed.*

4. View Tutorials on Internet Usage
***Description**: Offers step-by-step learning for digital literacy.*

- *Steps:*
  1. *User selects "Tutorials" section.*
  2. *Views tutorials by category (YouTube, WhatsApp, etc.).*
  3. *Watches videos.*

5. Caregiver Support
***Description**: Caregiver assists in setting up reminders or browsing services.*

- *Steps:*
  1. *Caregiver logs in with linked account.*
  2. *Views senior's remainders and notifications.*
  3. *Edits reminders.*
  4. *Logs out after support.*

6. Receive Notifications
***Description**: Sends timely alerts related to reminders or government deadlines.*

- *Steps:*
  1. *System checks for scheduled reminders or policy updates.*
  2. *Generates alert.*
  3. *User acknowledges or dismisses the alert.*

3 Design Overview

The software design of Oldie2GenZ is based on a Monolithic Architecture. In this approach, the entire application—including authentication,
reminders, notifications, tutorial delivery, government API integration, and user data management—is developed, deployed, and maintained as a
single cohesive unit.All modules are tightly integrated and operate within the same codebase and runtime environment. Rather than being broken
into separate service communicating via an API gateway, all internal communication happens through direct function or module calls within the
monolith. This monolithic approach follows the COMET methodology and uses UML diagrams such as use case, class, activity, and state diagrams
to model system behavior and structure.
The design emphasizes:
Simplicity in development and deployment, especially suitable for small to medium-scale applications like this.
Unified data access, since all components share a common database and code structure.
Faster internal communication, as no network calls are required between services.

Consistency in state and session management, which is easier to maintain in a single-unit system.

3.1 Design Goals and Constraints

Design Goals:

1. *Accessibility: Create a senior-friendly interface with large buttons, readable text.*
2. *Simplified Digital Access: Bridge the digital divide by providing easy access to government services and online platforms*
3. *Health Management: Implement reliable reminder systems for medications and appointments*
4. *Digital Literacy: Provide step-by-step tutorials for common digital tasks*
5. *Caregiver Integration: Allow caregivers to assist seniors remotely with account management*

Design Constraints:

1. *COMET Methodology: Must follow Concurrent Object Modeling and Architectural Design Method*
2. *UML Compliance: All system diagrams must use Unified Modeling Language standards*
3. *Hardware Limitations: Must function efficiently on low-end devices*
4. *Security Requirements: Implement AES-256 encryption and HTTPS for all data transmission*
5. *Accessibility Standards: Must comply with WCAG guidelines*
6. *Performance Requirements: Pages must load within 3 seconds, notifications within 1 second delay*

3.2 Design Assumptions

1. ***Stable Internet Connection**: Users will have reliable internet access for core functionality*
2. ***Basic Digital Literacy**: Users have minimal ability to interact with web interfaces*
3. ***Caregiver Support**: Many users will have caregivers to assist with initial setup*
4. ***Cross-Platform Compatibility**: Application will work on both desktop and mobile browsers*
5. ***Language Preferences**: Initial support for English, Hindi, and Telugu will suffice for most users*
6. ***Senior-Specific Needs**: Interface must account for potential vision/hearing impairments and reduced motor skills*

3.3 Significant Design Packages

1. ***User Management Package***
   - *Handles authentication (login/register)*
   - *Manages user profiles and preferences*
   - *Implements caregiver-senior relationships*
   - *Components: User, CareGiver, SeniorCitizen classes*
2. ***Reminder System Package***
   - *Manages medication and appointment reminders*
   - *Handles scheduling, notifications, and status tracking*
   - *Components: ReminderSystem, NotificationSystem classes*
   - *Subsystems: Scheduling, Alert Generation, Status Management*
3. ***Government Services Package***
   - *Provides access to government portals*
   - *Fetches and displays policy updates*
   - *Components: GovernmentService, ServiceManager classes*
4. ***Digital Literacy Package***
   - *Manages tutorials and learning modules*
   - *Tracks user progress*
   - *Components: Tutorials.*
   - *Subsystems: Content Delivery, Progress Tracking*
5. ***Interface Package***
   - *Handles all UI components*
   - *Manages accessibility features*
   - *Components: Web UI, High-Contrast Themes*
6. ***Database Package***

- *Stores all persistent data*
- *Components: MainDatabase, CacheDatabase*
- *Tables: Users, Reminders, Services*

Package Dependencies:

- *User Management depends on Database*
- *Reminder System depends on User Management and Notification*
- *Digital Literacy depends on Database and Interface*
- *Interface depends on all functional packages*

Layered Architecture:

1. ***Presentation Layer**: Web UI components*
2. ***Application Layer**: Core functionality packages*
3. ***Data Layer**: Database*
4. ***Infrastructure Layer**: Security, logging, and system services*

3.4 Dependent External Interfaces

The high-level design of the Oldie2GenZ application relies on several external interfaces to provide essential services such as government portal access, notifications, and learning content. These interfaces are integrated into specific internal modules to support core functionality like redirection, reminder alerts, and digital literacy.

The table below lists the public interfaces this design requires from other modules or applications.

External Application and Interface Name	**Module Using the Interface**	**Functionality/ Description**
Government Service Portals (e.g., Jeevan Pramaan)	Government Service	Redirects users to government portals for services like pensions, healthcare, and policy updates.
Notification APIs	NotificationSystem	Sends reminders (medicine, appointments)
Database	MainDatabase	Stores user profiles, reminders, preferences, and tutorial progress.
Security/Authentication Modules (e.g., OTP Services)	User (Authentication)	Validates user credentials during login/registration. Ensures secure access to personalized features.
Digital Literacy Content Providers	DigitalLiteracyModule	Fetches and displays tutorial videos/content

3.5 Implemented Application External Interfaces (and SOA web services)

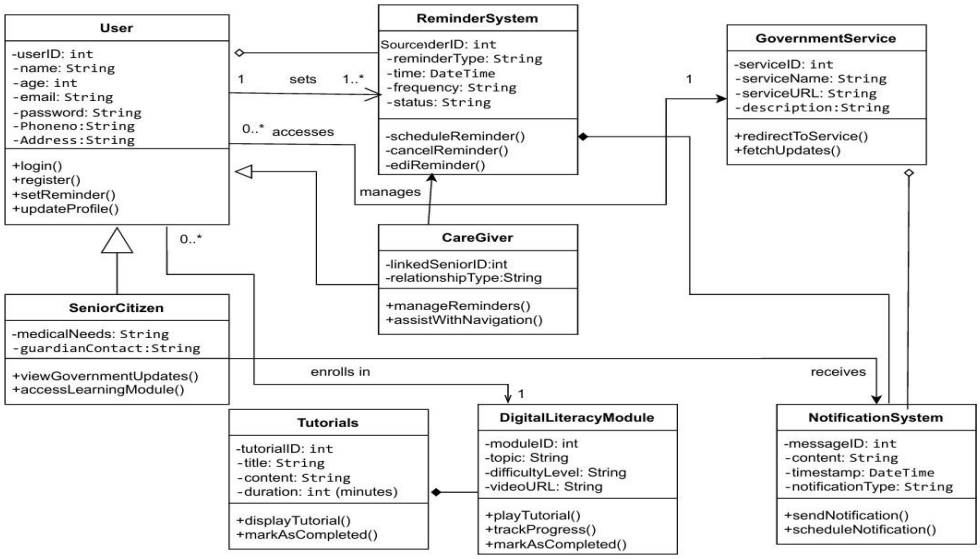
[The high level design identified interfaces that are provided or owned by the application. This section indicates which internal modules are responsible for implementing these interfaces.]

The table below lists the implementation of public interfaces this design makes available for other applications.

Interface Name	Module Implementing the Interface	Functionality/ Description
User Authentication	User Module	Implements login(), SignUp() and updateProfile() methods to handle user credentials and profile management. Uses secure encryption (AES-256) for passwords.
Reminder System	ReminderSystem Module	Implements scheduleReminder(), cancelReminder(), and editReminder() methods. Integrates with NotificationSystem to trigger text alerts at scheduled times.
Digital Literacy	DigitalLiteracyModule	Implements playTutorial(), playEntertainment().
Notification System	NotificationSystem Module	Implements sendNotification() and scheduleNotification() methods.
Database Integration	MainDatabase Module	Implements CRUD operations for user data, reminders, and tutorial progress.

4 Logical View

This section describes the internal design and structure of the Oldie2GenZ application. The system is divided into modules based on key functionalities like user authentication, reminders, digital learning, and access to government services. Each module is further broken down into classes that collaborate to implement the system's core behavior.



4.1 Design Model

The system is decomposed into the following modules, each represented by key classes and their responsibilities:

Module 1: User Management

- ***Classes:***
  - *User (Base Class)*
    - ***Attributes:*** *name, phoneNo.,password.*
    - ***Responsibilities:*** *Handles user registration, login, and profile updates.*
  - *SeniorCitizen (Subclass of User)*
    - ***Attributes:*** *medicalNeeds, guardianContact.*
    - ***Responsibilities:*** *Manages access to government services and learning modules.*
  - *CareGiver (Subclass of User)*
    - ***Attributes:*** *linkedSeniorID.*
    - ***Responsibilities:*** *Assists seniors with reminders and navigation.*

Module 2: Reminder System

- ***Classes:***
  - *ReminderSystem*
    - ***Attributes:****Name, Date, No. of Days, Time,Dosage .*
    - ***Responsibilities:*** *Schedules, cancels, and edits reminders.*
  - *NotificationSystem*
    - ***Attributes:*** *Content, timestamp, notificationType.*
    - ***Responsibilities:*** *Sends text alerts and updates reminder status.*

Module 3: Government Services

- ***Class:***
  - *GovernmentService*
    - ***Attributes:*** *ServiceName, serviceURL, description.*
    - ***Responsibilities:*** *Redirects to government portals and fetches updates.*

Module 4: Digital Literacy

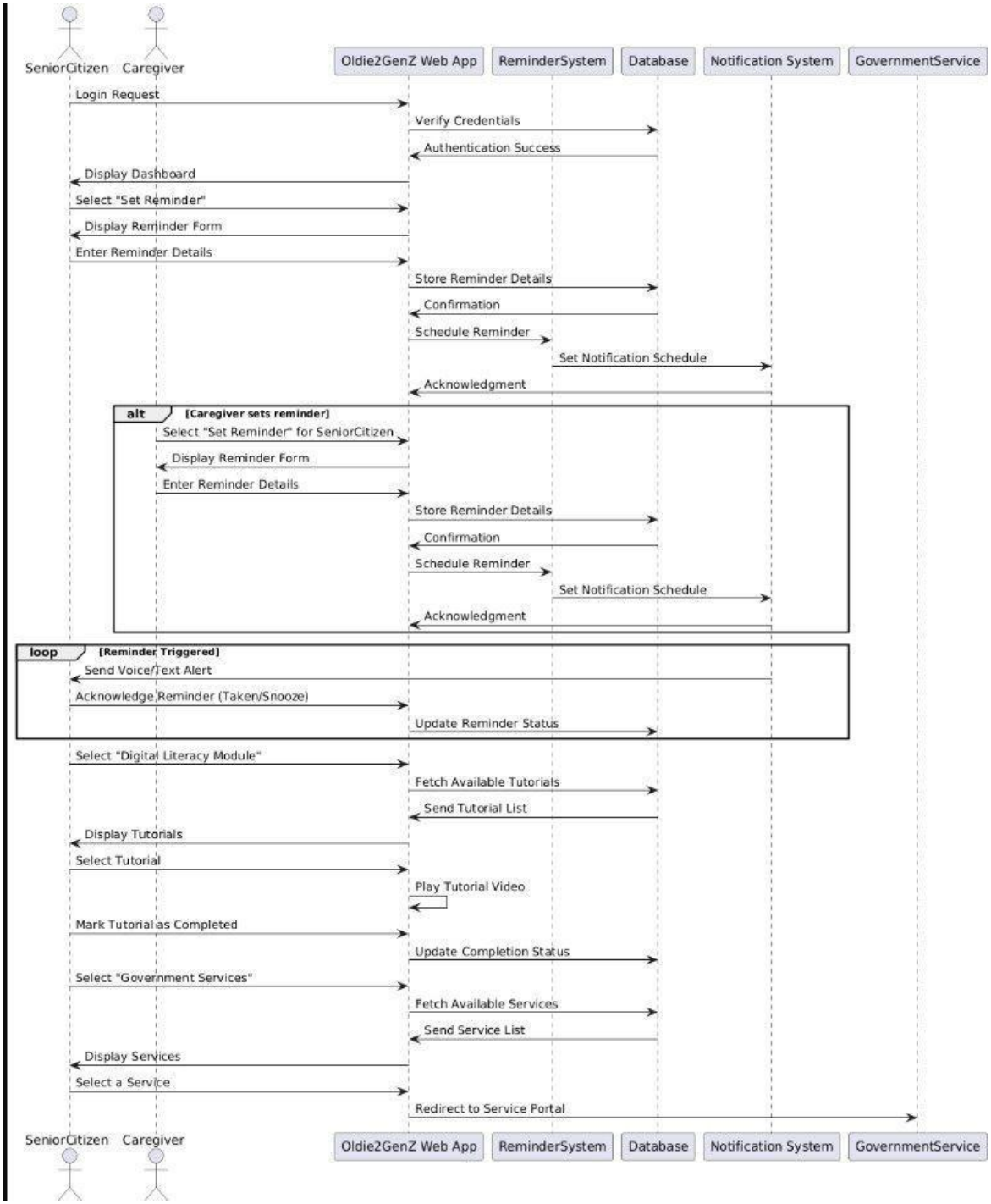
- ***Classes:***
  - *Tutorials*
    - ***Attributes:*** *tutorialID, title, content, duration.*
    - ***Responsibilities:*** *Displays tutorials.*

Relationships:

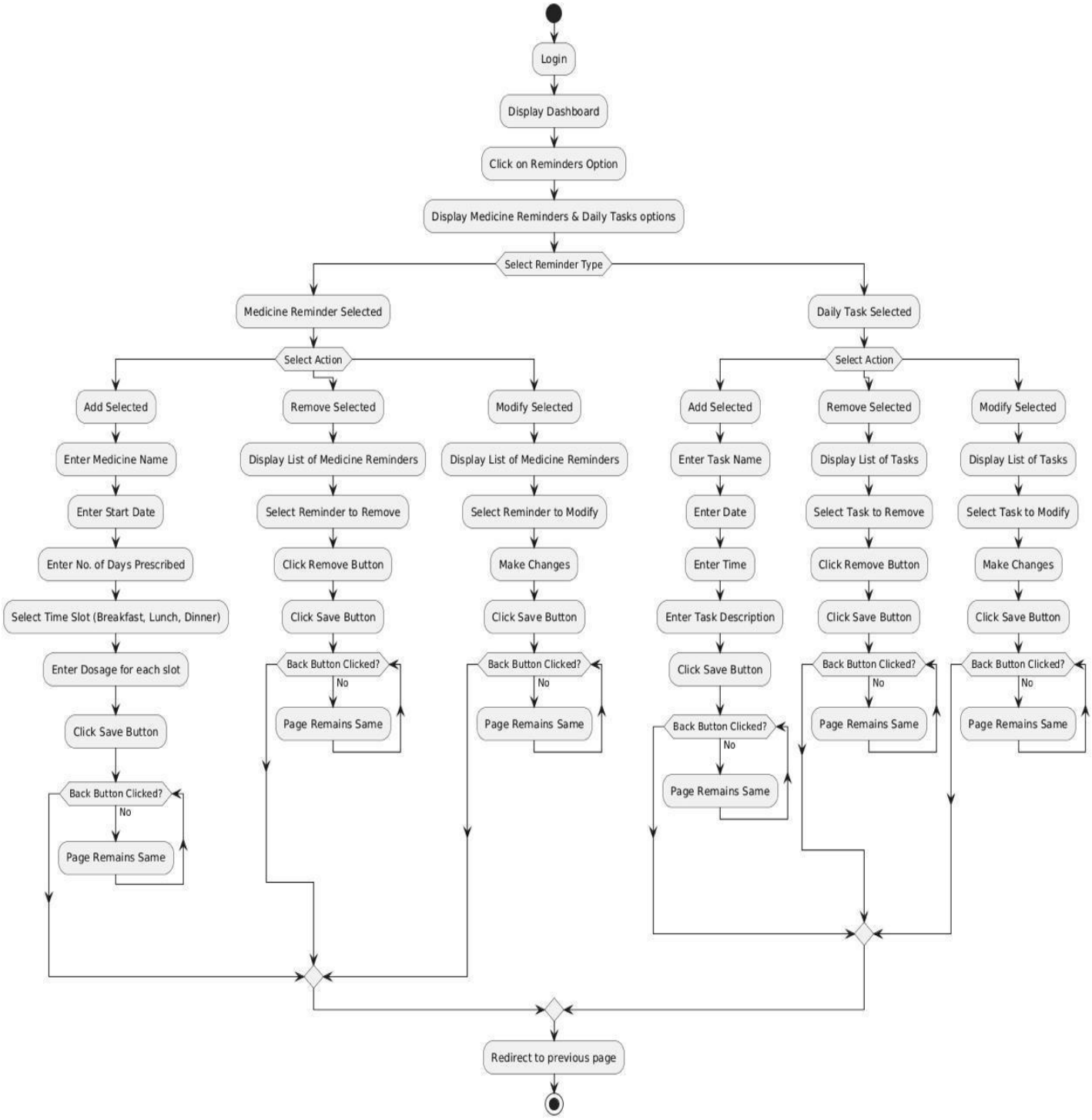
- *User aggregates ReminderSystem and accesses GovernmentService/DigitalLiteracyModule.*
- *CareGiver manages ReminderSystem for SeniorCitizen.*
- *NotificationSystem receives triggers from ReminderSystem.*

4.2 Use Case Realization

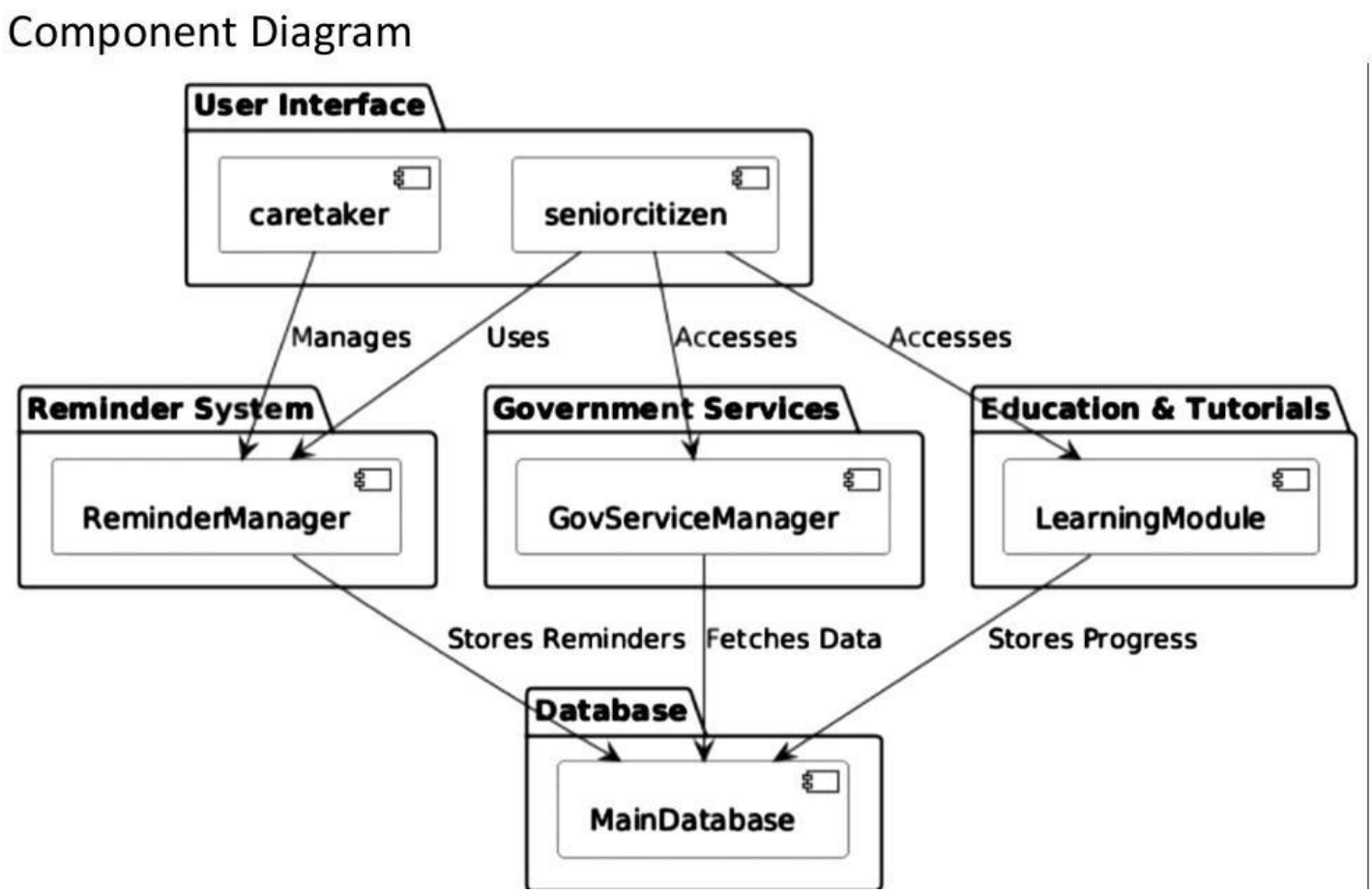
Sequence Diagram:



Activity Diagram:


Login
Display Dashboard
Click on Reminders Option
Display Medicine Reminders & Daily Tasks options
Select Reminder Type
Medicine Reminder Selected
Daily Task Selected
Select Action
Add Selected
Enter Medicine Name
Enter Start Date
Enter No. of Days Prescribed
Select Time Slot (Breakfast, Lunch, Dinner)
Enter Dosage for each slot
Click Save Button
Back Button Clicked?
No
Page Remains Same
Remove Selected
Display List of Medicine Reminders
Select Reminder to Remove
Click Remove Button
Click Save Button
Back Button Clicked?
No
Page Remains Same
Modify Selected
Display List of Medicine Reminders
Select Reminder to Modify
Make Changes
Click Save Button
Back Button Clicked?
No
Page Remains Same
Select Action
Add Selected
Enter Task Name
Enter Date
Enter Time
Enter Task Description
Click Save Button
Back Button Clicked?
No
Page Remains Same
Remove Selected
Display List of Tasks
Select Task to Remove
Click Remove Button
Click Save Button
Back Button Clicked?
No
Page Remains Same
Modify Selected
Display List of Tasks
Select Task to Modify
Make Changes
Click Save Button
Back Button Clicked?
No
Page Remains Same
Redirect to previous page

Component Diagram:



5 Data View

Oldie2GenZ uses persistent data storage to manage user profiles, caregiver relationships, reminders, tutorials, and entertainment. This ensures a personalized and consistent experience for each user. The system uses structured data models to maintain integrity and enable dynamic user interactions.

5.1 Domain Model

The Domain Model includes the following core entities:

- ***User**: Represents either an elderly user or a caregiver. A user can create reminders, access tutorials.*
- ***Reminder**: Stores scheduled alerts for users to manage medications or tasks.*
- ***Tutorial**: Provides educational content in text.*
- ***Service**: Links users to external useful services like healthcare or government schemes.*
- ***Caregiver**: Linked to elderly users for assistance.*
- ***Feedback**: Stores user opinions, suggestions, and ratings.*

Relationships:

- *A **User** can have multiple **Reminders** entries.*
- *A **Caregiver** is associated with one or more **Users**.*

- *Tutorials and Services are accessible to all users.*

5.2 Data Model (persistent data view)

The system uses a relational database to store persistent data. Each domain entity maps to a table:

- *Users: Stores user profile data*
- *Reminders: Linked to Users via SC_seq*
- *Services: Contains metadata about external services*
- *Caregivers: Associated with elderly users*

These tables are linked using foreign keys to ensure referential integrity.

5.2.1 Data Dictionary

Attribute Name	Data Type	Description
SC_mob	**Varchar**	**Unique identifier for each user**
SC_name	**Varchar**	**Full name of the user**
SC_password	**Varchar**	**Hashed password for login**
SC_seq	**int**	**Sequence of numbers to the SC**
Med_time	**TIME**	**Timestamp when the user was created**

6 Exception Handling

In the Oldie2GenZ application, we understand that our primary users (senior citizens) may not be comfortable with technical error messages or sudden system failures. So, we've designed our exception handling system to be user-friendly, non-technical, and informative. Behind the scenes, the system is also built to log and track every error to ensure smooth functioning and fast recovery.

Here's how we handle different types of errors:

1. Login Issues

If a user enters the wrong phone no. or password, the system gently notifies them:

"Login Failed Please try again."

After a few failed attempts, the user is guided to reset their password. All failed attempts are logged securely so that any suspicious activity can be tracked.

2. Reminder Errors

If someone tries to set a reminder in the past or enters an invalid time, the system highlights the mistake and helps them correct it with simple instructions like:

"Please select a valid future time for your reminder."

3. Unauthorized Access by Caregivers

Caregivers can assist seniors, but they're limited to certain features. If a caregiver tries to access a restricted

section, they're redirected with a message:
"You do not have permission to access this section."
Such attempts are logged for security.

4. External Service Failures (like Jeevan Pramaan or SMS issues)

Sometimes, government services or SMS gateways may be temporarily down. In such cases, the user sees a calm message:
"This service is temporarily unavailable. Please try again in a while."
Meanwhile, the system automatically tries again or switches to a backup method, like sending an SMS instead of an in-app alert.

5. Form Mistakes

If a user forgets to fill a required field or types something incorrectly, they are gently reminded right on the screen:
"Please fill all required fields correctly."
The system won't move forward until the issue is fixed, but it explains clearly what went wrong.

6. Something Unexpected Happens

If there's a server crash or some unexpected error, the user won't see a scary error page. Instead, we show a polite message:
"Something went wrong. We're fixing it! Please try again soon."
Meanwhile, our system logs all the technical details in the background and notifies the technical team if needed.

Behind the Scenes Logging & Alerts

Every exception is logged with a time, type of error, and (if available) the user's ID. Critical issues immediately alert the admin or support team. This way, we ensure no issue goes unnoticed, and the system gets fixed before it causes more trouble.
In short, we've made sure that our exception handling is both ***senior-friendly*** *and* ***technically robust****, so our users feel confident and supported while using the platform.*

7 Configurable Parameters

Oldie2GenZ*,* ***configurable parameters*** *refer to the settings and values that can be* ***adjusted without changing the codebase****, either by users (like senior citizens or their caregivers), system administrators, or developers through external configuration files, UI controls, or admin panels.*
Configurable Parameters in Oldie2GenZ
User Authentication & Personalization

- *Password complexity rules (length, special characters, etc.)*
- *User roles & permissions (senior, caregiver, admin)*

Health & Appointment Reminder System

- *Reminder frequency (daily, weekly, etc.)*
- *Reminder time (user-selectable)*
- *Snooze/repeat options for reminders*

Digital Literacy Tutorials

- *Tutorial playback speed*
- *Language options*
- *Text size in tutorials*

Notifications and Alerts

- *Enable/disable real-time policy updates*
- *Time window for receiving notifications*

User Interface Preferences

- *Font size and theme (dark/light mode)*

This table describes the simple configurable parameters (name / value pairs).

Configuration Parameter Name	Definition and Usage	Dynamic?
ReminderFrequency	Defines how often reminders (medication, appointments) are repeated (e.g., daily, weekly).	Yes
NotificationType	Specifies the mode of notification (voice, text, or both) for reminders and updates.	Yes
Font Size	Adjustable text size to improve readability for elderly users.	Yes
Language Preference	Allows users to choose their preferred language for tutorials and navigation.	Yes
Caregiver Access Level	Permissions assigned to caregivers for managing reminders/settings for elderly users.	Yes
EncryptionMethod	Specifies the encryption standard (AES-256) for user data storage and transmission.	No
Default Home Page Section	Let's the user choose which section (Reminders, Tutorials, etc.) opens first upon login.	Yes
User Role Permissions	Admin-controlled access levels (e.g., elderly user, caregiver, super admin).	Yes

8 Quality of Service

*This section outlines the design aspects of the **Oldie2Genz** website related to availability, security, performance, and monitoring. These aspects ensure that the application delivers a reliable, secure, and responsive experience to users of all ages, especially focusing on accessibility and ease of use for senior citizens.*

8.1 Availability

Oldie2GenZ is designed to maintain 99.9% uptime and ensure continuous service availability. Key design elements include:

- *Automatic recovery mechanisms for critical services such as reminder and notification modules.*
- *Scheduled database backups every 24 hours to prevent data loss.*
- *Light-weight frontend optimized for low-end devices and slower internet connections.*
- *No downtime during normal updates; system maintenance is scheduled during off-peak hours.*

8.2 Security and Authorization

Security is a top priority to protect senior users and their data. The system includes:

- *Password-based login system with encrypted password storage using hashing (e.g., bcrypt or SHA-256).*
- *All data transmissions use HTTPS to protect against data interception.*
- *Role-based access control (RBAC) to limit caregiver permissions to reminder and support sections only.*
- *Secure session management with auto-expiry after inactivity.*
- *Users are notified via email whenever a caregiver makes changes on their behalf.*
- *Administrative access is limited to authorized backend staff, and all admin actions are logged.*

8.3 Load and Performance Implications

The system is built to handle expected loads while ensuring smooth performance:

- *Supports 100+ concurrent users with minimal performance degradation.*
- *Dashboard and main pages load within 3 seconds on 4G internet.*
- *Reminder and notification APIs process within 1 second under normal conditions.*
- *Video tutorials load with a maximum delay of 2 seconds and buffer efficiently.*
- *Database is indexed for faster queries on user reminders, preferences, and government updates.*
- *Regular performance testing is conducted to simulate user behavior and improve response time.*

8.4 Monitoring and Control

Monitoring tools and health checks are integrated into the backend to maintain system stability:

- *Monolithic (e.g., notification, reminder, tutorial) report their health status periodically.*
- *Logs are maintained for login attempts, reminder creation, notification status, and caregiver activity.*
- *Real-time alerts are triggered in case of API failures, missed reminders, or login anomalies.*
- *Admin panel displays system usage metrics, such as:*
    - *Active users*
    - *Reminder delivery success rate*
    - *System uptime*
- *Automated scripts restart failed services and ensure background jobs (like daily reminder checks) run on time.*