Software Design Specifications

for

AgeWise Version: - 1

Prepared by:

Chetan Naidu Roll no- SE22UARI137
Rahul Kaja Roll no: - SE22UARI139
A Nandan Roll no: - SE22UARi001
Sivamani Roll no: - SE22UARI119
Goutham Roll no: - SE22UARI010

Team Marscode(Team 30)

Document Information

Title: AgeWise Software Design specification document	
	Document Version No:1
	Document Version Date: 08-04-25
Prepared By: Team Marscode	
	Preparation Date: 08-04-25

Version History

Ver. No.	Ver. Date	Revised by	Description	Filename
1	**08-04-25**	MarsCode	This Software Design Specification (SDS) outlines the architecture, components, and implementation strategies for the AgeWise web platform. AgeWise is a user-friendly solution designed to assist senior citizens in accessing government services, managing reminders, receiving news updates, and learning digital skills with the help of AI-based guidance and guardian support. This document serves as a blueprint for developers, testers, and stakeholders to implement a secure, scalable, and accessible system	**AgeWise_Software_Design_Specification**

Table of Contents

1. Introduction

The Software Design Specification (SDS) document for AgeWise outlines the comprehensive design and architectural blueprint for a web-based platform that enables senior citizens to access essential digital services with ease. Developed in accordance with IEEE software engineering standards, this document translates the functional and non-functional requirements defined in the Software Requirements Specification (SRS) into an actionable design plan for implementation. AgeWise aims to bridge the digital divide for elderly users by offering an intuitive platform equipped with features like government service redirection, medication reminders, digital literacy tutorials, AI-powered chatbot assistance, and voice navigation. The platform is developed with a strong focus on accessibility, security, scalability, and user-friendliness to support users with varying levels of digital literacy and physical ability.

1.1 Purpose

The purpose of this document is to provide a detailed description of the software architecture and design decisions for the AgeWise system. It serves as a guide for developers, testers, project managers, and other stakeholders involved in the software development lifecycle. This document ensures that the system's design meets the outlined requirements and allows for consistent, maintainable, and scalable implementation.
The SDS includes component-level designs, module responsibilities, data interactions, interface definitions, use case realizations, exception handling mechanisms, and quality of service specifications. Each section is structured to support implementation clarity, alignment with stakeholder needs, and readiness for future enhancements.

1.2 Scope

This document applies to the design of the AgeWise Web Application, version 1.0. It focuses on the architectural structure, interface interactions, class responsibilities, and module communication mechanisms. The scope includes user and admin functionality, integration with third-party APIs (e.g., government portals, notification services), and accessibility-driven UI features.
The SDS supports implementation efforts for:
User registration and authentication
Government service access and guidance
Health and medication reminder systems
Digital literacy tutorials
AI chatbot and voice assistance
Admin and guardian-supported operations

1.3 Definitions, Acronyms, and Abbreviations

A comprehensive list of terms, abbreviations, and acronyms is available in the SRS (Section 1.4). Key terms include:
AI: Artificial Intelligence
RBAC: Role-Based Access Control
WCAG: Web Content Accessibility Guidelines
OAuth: Open Authorization protocol
MFA: Multi-Factor Authentication
TLS: Transport Layer Security
TTS: Text-to-Speech

1.4 References

Software Requirements Specification for AgeWise, Version 1.0, MarsCode, March 2025
IEEE 1016-2009: IEEE Standard for Information Technology—Systems Design—Software Design Descriptions
WCAG 2.1 Guidelines: https://www.w3.org/TR/WCAG21/
MongoDB Documentation: https://www.mongodb.com/docs/
React.js Documentation: https://react.dev/

OAuth 2.0: https://oauth.net/2/

2 Use Case View

This section outlines the primary use cases of the AgeWise platform, each reflecting a central aspect of system functionality. These use cases were selected based on their significance in supporting elderly users' digital independence and promoting usability, accessibility, and safety. Each use case includes the actors, triggers, workflows, and exceptions relevant to its realization in the software design.

2.1 Use Cases

Use Case 1: Set and Receive Reminders

Actors
- Primary Senior Citizen
- Secondary: Guardian, Notification System

Brief Description
- Enables users or their guardians to set health-related or government task reminders. The system ensures delivery of timely notifications via SMS, email, or voice alerts.

Preconditions:
- The user must be logged into the system.
- Notification preferences should be configured.

Main Flow:
- User navigates to the "Reminders" section.
- Selects "Set New Reminder".
- Inputs reminder details (title, time, frequency).
- Reminder is saved in the database.
- The notification system dispatches alerts at the scheduled time.

Alternate Flow:
- If a guardian is setting the reminder, they first select the relevant user profile.

Exceptions:
- No internet connection: The reminder is saved locally and synchronized later.
- Invalid time entry: The system prompts for correction.

Includes:
- Use Case: User Authentication

Use Case 2: Access Government Services

Actors:
- Primary: Senior Citizen
- External: Government Services API

Brief Description:
- Provides one-click access to government services such as Jeevan Pramaan and pension verification with guided instructions.

Preconditions:
- User is authenticated.
- API connectivity is established.

Main Flow:
- User selects "Government Services" from the dashboard.
- System displays available government services.
- User chooses a service and is redirected to the official portal.

- Step-by-step guidance is provided alongside.

Alternate Flow:
- If the government portal is down, fallback instructions are displayed.

Exceptions:
- Authentication timeout: Prompt the user to log in again.
- API error: Display error message and support contact.

Includes:
- Use Case: Secure Authentication

Use Case 3: Access News and Updates

Actors:
- Primary: Senior Citizen
- External: News API

Brief Description:
- Allows users to stay updated with simplified, curated news content available in regional languages, with voice readout functionality.

Preconditions:
- User is authenticated.
- Internet connectivity is available.

Main Flow:
- User navigates to the "News" section.
- System fetches and displays articles using a third-party News API.
- Content is shown in a high-contrast, large-font layout.
- User can read or use text-to-speech functionality.

Alternate Flow:
- User selects preferred language for translation.

Exceptions:
- API unresponsive: Display last cached news or error message.
- Translation error: Revert to default English content.

Includes:
- Use Case: Text-to-Speech
- Use Case: Multi-Language Support

Use Case 4: Watch Tutorials and Learn Digital Skills

Actors:
- Primary: Senior Citizen
- System Component: Tutorial System

Brief Description:
- Provides access to educational tutorials on using digital tools like WhatsApp, Zoom, email, and browsing safely.

Preconditions:
- Tutorials are preloaded or downloadable.
- User is logged in.

Main Flow:
- User selects "Digital Tutorials" from the main menu.
- List of topics is displayed.
- User selects a topic to begin.
- Content is presented via video or guided steps.

Alternate Flow:
- Tutorials can be downloaded for offline viewing.

Exceptions:

- No internet: Prompt user to download tutorial packs.
- Tutorial not found: Suggest alternative options.

Includes:
- Use Case: AI Chatbot Support

Use Case 5: AI Chatbot Support for Instant Help
Actors:
- Primary: Senior Citizen
- System Component: AI Chatbot

Brief Description:
- An AI-powered chatbot guides users through platform features and helps troubleshoot common issues using natural language processing.

Preconditions:
- User is authenticated.
- Chatbot services are operational.

Main Flow:
- User selects “Help” or activates voice assistance.
- Chatbot greets and prompts the user.
- User asks a question (via voice or text).
- Chatbot replies with step-by-step instructions or performs the task.

Alternate Flow:
- For complex queries, chatbot escalates to human support or suggests contacting a guardian.

Exceptions:
- Voice not recognized: Prompt user to enter input via text.
- Chatbot not available: Display FAQ section.

Includes:
- Use Case: Voice Input and Output

3. Design Overview

This section provides a comprehensive view of the AgeWise system architecture, major design components, external dependencies, and the guiding assumptions behind the solution architecture. The design focuses on creating an accessible, modular, scalable, and secure system aligned with user needs, especially catering to elderly users with limited digital literacy.

3.1 Design Goals and Constraints

Design Goals:

- **Accessibility First**: Fully compliant with WCAG 2.1 guidelines to support users with visual, auditory, or motor impairments.
- **Simplicity and Usability**: Large touch targets, guided navigation, and minimal decision-making steps.
- **Security and Privacy**: End-to-end TLS 1.3 encryption, secure authentication (OAuth/OTP), GDPR and Indian IT Act compliance.
- **Scalability and Reliability**: Target 99.9% uptime, horizontal scaling, and load balancing to support up to 5000 users.
- **Multi-language Support**: Regional language toggle with high translation accuracy (>95%) and text-to-speech support.

Design Constraints:

- Must function on low-spec devices (minimum 2GB RAM, 1GHz CPU).
- Web pages must load within 3 seconds on 3G networks.
- Third-party APIs (e.g., government services) may be unreliable or rate limited.
- Must support offline access for reminders and tutorials.

3.2 Design Assumptions

- Senior users will have assistance during initial onboarding.
- Guardians will be available to help manage tasks such as reminders or account recovery.
- Internet access is generally stable, but the system includes offline fallback for key features.
- Government APIs and third-party services (Twilio, Firebase) are mostly available and responsive.
- A microservice architecture is assumed to allow modular and future mobile app development.

3.3 Significant Design Packages

Package Name	Responsibility
`auth-service`	Manages registration, login, OTP/MFA, and role-based access control (RBAC).
`reminder-service`	Handles creation, scheduling, and notification of medication/government reminders.
`gov-access-service`	Connects to external government portals with redirection and fallback mechanisms.
`tutorial-service`	Manages delivery of digital skill tutorials (text/video/offline downloads).
`news-service`	Fetches localized news articles; supports multi-language and TTS rendering.
`chatbot-engine`	AI chatbot to assist with FAQs, onboarding, and general navigation queries.
`guardian-module`	Allows family members to manage reminders and monitor activity securely.
`admin-console`	Used by admins to approve users, moderate content, and monitor system analytics.

3.4 Dependent External Interfaces

External Module	Interface Description	Usage in AgeWise
Government Portals	REST APIs or redirect URLs	Pension verification, ID validation, welfare service access
Twilio / Firebase / SendGrid	Notification APIs	Delivery of SMS, email, and push reminders
Google OAuth / OTP Gateway	Secure Authentication	Multi-factor authentication for account login
NewsAPI or Equivalent	REST API	Display curated news headlines in simplified format
AWS / Heroku / Render	Cloud Hosting	Deployment, scaling, and uptime reliability

3.5 Implemented Application External Interfaces

Interface Name	Implementing Module	Functionality Description
/api/reminders	reminder-service	Create, update, delete, and notify about health or task reminders
/api/gov-services	gov-access-service	List available government services and redirect users accordingly
/api/tutorials	tutorial-service	Retrieve digital literacy content by category or topic
/api/news	news-service	Provide daily news updates in multiple formats and languages
/api/chatbot	chatbot-engine	NLP-based chatbot responses for help and instructions
/api/guardian	guardian-module	Manage linked accounts, reminders, and user activity tracking
/api/admin	admin-console	Access for system monitoring, content moderation, and account control

4. Logical View

This section presents the detailed internal architecture of the AgeWise platform. It outlines how the system's components and classes interact to support the key use cases while adhering to principles of modularity, maintainability, accessibility, and performance. The software architecture is designed using the COMET methodology and follows an object-oriented approach to promote extensibility and simplicity.

4.1 Design Model

The AgeWise system adopts a layered architecture composed of the following:

1. Presentation Layer

- Built using React.js with accessibility-first UI design.
- Provides a clean, responsive interface with large buttons, multi-language toggles, and text-to-speech support.
- Communicates with backend services through REST APIs.

2. Application Layer

- Coordinates user workflows such as reminder scheduling, guardian access, and chatbot queries.
- Handles request routing, session control, and validation logic.

3. Service Layer

- Implements domain-specific business logic for features like reminders, news, and tutorials.
- Facilitates interactions between the application layer and the data persistence layer.

4. Persistence Layer

- Utilizes MongoDB for unstructured data (e.g., user profiles, reminders).
- Uses PostgreSQL for structured datasets such as logs, service activity, and admin analytics.

5. Security Layer

- Provides OAuth 2.0 and OTP-based authentication mechanisms.
- Implements TLS 1.3 encryption and role-based access control (RBAC) for data protection.

Key Classes and Responsibilities

Class Name	Responsibility
`ReminderManager`	Creates, updates, and schedules user and guardian-set reminders.
`NewsAggregator`	Fetches and filters articles from third-party APIs and handles translation.
`TutorialService`	Retrieves and displays interactive tutorials; manages offline content.
`GuardianController`	Manages linked guardian profiles and permissioned actions.
`ChatbotEngine`	Responds to user queries via NLP-based flow and step-by-step help.
`NotificationDispatcher`	Sends notifications via SMS, email, and push channels using third-party APIs.

Each class is developed to maintain single-responsibility, support extensibility, and align with modular development practices.

4.2 Use Case Realization

The following describes how the system realizes each key use case internally using its class and module structure.

Use Case: Set and Receive Reminders

- **Modules Involved**: `ReminderManager`, `UserController`, `NotificationDispatcher`, `GuardianController`
- **Flow**:
    - User submits reminder details via the UI.
    - The data is processed and stored through the `ReminderManager`.
    - The system schedules notifications and dispatches alerts at the set time.
    - If missed, the guardian is notified automatically.

Use Case: Access Government Services

- **Modules Involved**: `GovServiceRouter`, `UserController`, `Logger`
- **Flow**:
    - User selects a service from the dashboard.
    - System retrieves service info via `GovServiceRouter`.
    - User is redirected to the respective portal with instructional overlays.
    - The interaction is logged.

Use Case: Access News and Updates

- **Modules Involved**: `NewsAggregator`, `TranslationService`, `TTSService`
- **Flow**:
    - User opens the "News" section.
    - `NewsAggregator` fetches the latest content from external APIs.
    - Articles are translated if needed and rendered.
    - The user listens to the content via `TTSService`.

Use Case: AI Chatbot Support

- **Modules Involved**: `ChatbotEngine`, `NLPProcessor`, `HelpContentService`
- **Flow**:
    - User activates the chatbot by clicking "Help".
    - `ChatbotEngine` receives and processes the input.
    - The system returns step-by-step instructions or redirects the user to relevant modules.

Use Case: Guardian-Assisted Management

- **Modules Involved**: `GuardianController`, `ReminderManager`, `UserProfileService`
- **Flow**:
    - Guardian logs into their account and accesses the linked senior profile.
    - Reviews logs, updates reminders, and adjusts settings as needed.

- Changes are saved and relevant parties are notified.

This logical structure ensures that the AgeWise system remains organized, intuitive, and optimized for both functionality and accessibility. Each module interacts through clean, well-defined interfaces, making the platform scalable and maintainable.

5. Data View

This section describes the persistent data storage perspective of the AgeWise system. It outlines the domain model, data relationships, and key data elements used throughout the platform. The AgeWise system employs a hybrid data storage strategy using **MongoDB** for flexible, unstructured documents (such as user profiles and reminders) and **PostgreSQL** for structured relational data (such as system logs and analytics)**.**

5.1 Domain Model

The domain model captures the key entities and relationships required by the system's core modules. Each entity reflects a specific function such as reminders, user management, chatbot queries, and guardian assistance.

Key Entities:

- **User**
  - Attributes: `userID`, `name`, `dob`, `languagePreference`, `role`, `email`, `phoneNumber`, `authMethod`
  - Description: Represents both senior users and guardians. Stores authentication and personal preference details.
- **Reminder**
  - Attributes: `reminderID`, `userID`, `reminderType`, `reminderTime`, `frequency`, `status`
  - Description: Manages all reminders set for medications, appointments, and government deadlines.
- **GuardianLink**
  - Attributes: `guardianID`, `linkedUserID`, `permissions`
  - Description: Stores the relationship and access permissions between guardians and senior users.
- **Notification**
  - Attributes: `notificationID`, `userID`, `message`, `timestamp`, `status`, `channel`
  - Description: Logs all system-generated alerts sent via SMS, email, or push notification.
- **NewsArticle**
  - Attributes: `articleID`, `title`, `content`, `language`, `publishedDate`, `source`
  - Description: Stores curated articles retrieved from external news APIs with support for translation.
- **ChatbotQuery**
  - Attributes: `queryID`, `userID`, `queryText`, `timestamp`, `responseGiven`
  - Description: Tracks user interactions with the chatbot including query logs and responses.
- **AdminLog**
  - Attributes: `logID`, `adminID`, `action`, `targetUserID`, `timestamp`
  - Description: Tracks admin-level operations for monitoring and auditing purposes.

5.2 Data Model (Persistent Data View)

The data model reflects how the system stores, manages, and retrieves essential records across MongoDB and PostgreSQL. Each major entity persisted in collections or tables depending on structure and query needs.

5.2.1 Data Dictionary

Field Name	Type	Description	Entity

userID	String (UUID)	Unique identifier for a user (senior, guardian, or admin)	User
role	Enum	"SeniorCitizen", "Guardian", "Admin"	User
languagePreference	String	Preferred display language, e.g., "English", "Hindi", "Telugu"	User
reminderID	String (UUID)	Unique identifier for each reminder	Reminder
reminderTime	DateTime	Scheduled time when the reminder should be triggered	Reminder
status	Enum	"Pending", "Completed", "Missed"	Reminder, Notification
notificationType	Enum	Notification delivery mode: "SMS", "Email", "Push"	Notification
articleID	String	Unique identifier for a news article	NewsArticle
queryText	String	Text input sent by user to the chatbot	ChatbotQuery
responseGiven	String	Chatbot response text provided to the user	ChatbotQuery
logID	String	Unique log entry ID for administrative activity	AdminLog
action	String	Description of action taken by admin	AdminLog
authMethod	Enum	"OAuth", "OTP", "Password"	User
timestamp	DateTime	Records date and time of a specific event or interaction	All Entities

Each data object includes system-generated timestamps for creation and last update to support versioning, consistency, and compliance with data retention standards.

6. Exception Handling

This section outlines the exception handling strategy used in the AgeWise platform to ensure system reliability, user safety, and seamless user experience. The system is designed to detect, log, and respond gracefully to anticipated and unanticipated runtime errors. Exception handling is implemented consistently across all layers (frontend, backend, and API integration) and adheres to best practices for accessibility and user feedback.

6.1 Exception Categories

1. User Input Exceptions

- Triggered when users enter invalid or incomplete data in forms (e.g., reminder time not selected, invalid date of birth).
- **Resolution Strategy**:
    - Frontend validation with real-time error prompts.
    - Backend sanitization and validation.
    - Display of clear, accessible error messages (e.g., large font, voice feedback).

2. Authentication and Authorization Exceptions

- Occur when login fails, session expires, or unauthorized access is attempted.

- **Resolution Strategy**:
    - Redirect to login page with session timeout message.
    - Role-based access control enforcement (RBAC).
    - Logging of failed access attempts for audit purposes.

3. External API Failures

- Result from unresponsive or faulty government services, news APIs, or notification providers.
- **Resolution Strategy**:
    - Retry mechanism with exponential backoff (where applicable).
    - Fallback to cached data or display of "service temporarily unavailable" message.
    - User alerts if reminders or notifications fail to send.

4. Database Exceptions

- Include connection timeouts, write failures, or data format inconsistencies.
- **Resolution Strategy**:
    - Automatic rollback for failed transactions.
    - Error logging with timestamps and affected entity ID.
    - User feedback such as "Unable to save your changes at this time."

5. Network and Connectivity Errors

- Occur during slow internet conditions or offline access.
- **Resolution Strategy**:
    - Offline caching for tutorials and reminders.
    - Real-time connectivity status indicators.
    - Retry and sync data when connection is restored.

6. Chatbot and Voice Assistant Errors

- Triggered by unrecognized voice input or NLP misinterpretation.
- **Resolution Strategy**:
    - Prompt user to rephrase or switch to text input.
    - Offer help topics if input is ambiguous.
    - Redirect to human support or guardian notification after repeated failure.

6.2 Logging and Monitoring

All exceptions are centrally logged for analysis and debugging. The logging module captures:

- Timestamp of occurrence.
- User or session ID (if available).
- Module and function where the error occurred.
- Full stack trace for backend exceptions.

- Client device/environment details for frontend issues.

Logs are stored securely in compliance with **GDPR** and **Indian IT Act** requirements and retained for a rolling 90-day period. Sensitive user data is excluded from logs.

6.3 User Feedback and Accessibility

- All error messages follow accessibility standards: large font, high contrast, and voice narration where applicable.
- Errors are phrased in user-friendly language (e.g., "Something went wrong—please try again").
- For repeated issues, the system offers alternative actions or contact options.

6.4 Exception Handling Tools and Frameworks

- **Frontend**: React Error Boundaries, Toast Notifications, Formik/Yup for validation.
- **Backend**: Node.js error middleware, try/catch blocks, centralized exception handler.
- **APIs**: Axios interceptors for external API errors with fallback logic.

This robust exception handling design ensures that the AgeWise platform remains user-friendly, secure, and resilient—even in unpredictable conditions.

7. Configurable Parameters

This section defines the configurable parameters used across the AgeWise platform. These parameters allow for flexibility in adapting to different deployment environments, user preferences, third-party service limits, and performance tuning. Some parameters can be modified at runtime (dynamic), while others require a system restart or redeployment.

7.1 Configuration Strategy

AgeWise supports configuration through:

- **Environment variables** (for deployment-specific values such as API keys, ports).
- **Admin dashboard** (for updating non-sensitive, user-facing defaults such as reminder intervals).
- **Configuration files** (`config.js`, `.env`, `settings.yaml`) for structured and secure parameter management.

All sensitive configuration parameters are encrypted at rest and are only accessible by authorized components or administrators.

7.2 Configurable Parameters Table

Parameter Name	Definition and Usage	Dynamic?
DEFAULT_LANGUAGE	Sets the default language for new users (e.g., English, Hindi)	Yes
REMINDER_RETRY_LIMIT	Number of times a failed reminder notification should be retried	No
NEWS_REFRESH_INTERVAL	Frequency (in minutes) at which news articles are fetched from the API	Yes
TTS_ENABLED	Enables/disables text-to-speech across the platform	Yes
VOICE_ASSIST_TIMEOUT	Timeout duration (in seconds) for voice commands before fallback to text input	Yes
API_RESPONSE_TIMEOUT	Timeout for all outgoing API requests to external services (in milliseconds)	No
MAX_GUARDIAN_LINKS	Maximum number of guardian accounts a senior citizen can be linked with	No
NOTIFICATION_CHANNELS	Available modes of notification (SMS, Email, Push)	Yes
OFFLINE_TUTORIAL_SYNC	Enables syncing of tutorials for offline access	Yes
SYSTEM_LOG_RETENTION_DAYS	Number of days system logs are retained for auditing and monitoring	No
ENABLE_TWO_FACTOR_AUTH	Toggles multi-factor authentication using OTP	Yes
MAINTENANCE_MODE	Switches the platform into maintenance mode (alerts all users and restricts new logins)	Yes

7.3 Parameter Management and Access

- **System Admins** can update most parameters via the backend or admin dashboard.
- **Environment-based configurations** (e.g., API keys, database URIs) are only accessible through secure deployment pipelines and are never exposed to the frontend.
- Parameters that impact user experience (e.g., language, TTS, notification channels) are cached on the client side and updated when changed.

7.4 Security Considerations

- Sensitive parameters (e.g., API secrets, authentication keys) are stored in secure environment files and not hardcoded in the source code.
- Changes to critical configurations are logged and version controlled.
- Runtime dynamic parameters are validated before application to prevent misconfiguration and system failures.

This configuration system ensures that AgeWise remains flexible, secure, and adaptable to a variety of use cases, regions, and deployment environments.

8. Quality of Service

This section outlines the key quality attributes of the AgeWise platform, which ensure it meets expectations around availability, security, performance, and operational monitoring. These quality attributes are essential to delivering a reliable, user-friendly, and accessible experience for senior citizens, guardians, and administrators.

8.1 Availability

- The system is designed for **99.9% uptime**, ensuring that essential services such as reminders, tutorials, and chatbot assistance are always accessible.
- The platform is hosted on a **cloud infrastructure (AWS/Heroku/Render)** with auto-scaling and failover support.
- **Scheduled maintenance windows** are communicated to users in advance via notifications.
- **Redundant database backups** are created every 12 hours to prevent data loss.
- **Session persistence and recovery** mechanisms ensure that users can resume interrupted tasks after short downtimes.

8.2 Security and Authorization

- AgeWise enforces **OAuth 2.0** and **OTP-based multi-factor authentication (MFA)** for secure access.
- All sensitive user data is **encrypted using TLS 1.3 in transit** and **AES-256 at rest**.
- **Role-Based Access Control (RBAC)** ensures that only authorized users (senior citizens, guardians, admins) can access relevant modules and data.
- The platform complies with **GDPR** and the **Indian IT Act**, ensuring legal data privacy and protection standards are met.
- Failed login attempts are rate-limited, and suspicious activity (e.g., repeated failed logins) triggers automated account lockout and alerts.
- Admin activities are **logged with timestamps** for audit purposes.

8.3 Load and Performance Implications

- The system is designed to support at least **500 concurrent users at launch**, scalable to **5000 users** via cloud-based load balancing.
- **API response times** are maintained under **1 second** for most operations under normal load.
- **High-priority processes** (e.g., reminder dispatch, voice commands) are optimized to execute within **500ms**.

- **Caching mechanisms** are used to accelerate frequent data access (e.g., tutorials, dashboard content).
- News articles and tutorials are **pre-fetched and cached**, minimizing delays in content delivery.
- **Database indexing and query optimization** ensure fast access to structured records in PostgreSQL and MongoDB.

8.4 Monitoring and Control

- Real-time monitoring tools are used to track server uptime, API health, error rates, and user activity.
- The system exposes **key performance indicators (KPIs)** such as:
    - Server health (CPU, memory usage)
    - Daily active users
    - Reminder success/failure rates
    - Notification delivery times
- Admin dashboards include:
    - Live user tracking
    - Reminder queues and failure logs
    - System health overview
- Alerts are triggered in case of:
    - Server downtime
    - Excessive API error responses
    - Reminder dispatch failures
- All logs are retained for **90 days** and can be exported for compliance review or debugging.

The AgeWise platform is engineered with resilience, responsiveness, and security in mind to ensure continuous, safe, and empowering access for elderly users in digital environments.