Software Requirements Specification Document

# Product Perspective

TheReminder is a stand-alone android app that focuses on accessibility. The vibration based reminders that will be included in the app are not compatible with desktop environments therefore the project focuses on android mobile systems that have haptic systems. The app focuses on being accessible for a wide range of users including those with various impairments.

# Product Functions

TheReminder app will offer a simple and accessible app that allows users to organize their tasks.

* **Task Management**

Users can add, update, delete and save tasks.

* **Task Prioritization**

Users can assign a priority (high, low, medium etc.) to their tasks for better organization.

* **Reminder Type Selection Per Task**

Users can choose 1 or multiple reminder types per task that best accommodate their needs.

* **Profile Accessibility Management**

Users can configure various accessibility settings to allow a better user experience.

# User Characteristics

The app to provide an straightforward and easy way for all users to manage their tasks.The target users can be split into 2 general categories:

* **General users**

General users are the users that don’t have impairments. The app provides these users an efficient task management solution, along with an user interface that is easy to understand and navigate.

* **Users with disabilities**

The app offers accessibility options for various types of impairments:

* + **Users with hearing impairments**

Haptic reminders

Visual reminders

* + **Users with visual impairments**

High contrast ui options

* + **Users with motor impairments**

Resizable ui options.

# Constraints

The development and operation of *TheReminder* application are subject to the following technical and operational constraints:

* **Platform Limitation**:  
   The application is designed exclusively for Android mobile devices. This constraint is due to the reliance on hardware-specific features such as haptic (vibration) feedback, which are not universally supported across platforms like iOS or desktop systems.
* **Programming Language & Framework**:  
   The application will be developed using the Dart programming language within the Flutter framework. This choice supports cross-platform capabilities but will be constrained to Android deployment for this project.
* **Local Storage Requirement**:  
   Task data, user preferences, and reminder configurations will be stored locally using the SQLite database engine. Cloud synchronization and remote database access are intentionally excluded from the project scope.
* **Hardware Dependencies**:  
   The application requires devices with built-in support for vibration motors and audio output to deliver reminders effectively. Devices lacking these features may not support the full functionality of the app.
* **Accessibility Compliance**:  
   All user interface components must comply with basic accessibility guidelines, such as support for screen readers, high contrast modes, and scalable text. These considerations may limit design choices but are critical for usability.
* **Time Constraint**:  
   The final version of the application and all related documentation must be completed and submitted by the demanded deadline. No extensions are planned beyond this deadline.

# System Features

### **1. Reminder Type Configuration**

Allows users to select one or multiple reminder types (audio, vibration, visual) per task. This configuration ensures that users with different accessibility needs can tailor their reminder experience accordingly.

* Accessible through the Add/Edit Task screen
* Strategies applied at runtime for flexible behavior
* Validates user input and provides fallbacks (e.g., default reminder)

### **2. Reminder Trigger System**

Manages the automatic triggering of reminders based on task deadlines.

* Uses a background timer to track scheduled reminders
* Sends appropriate notifications using the selected reminder strategies
* Supports multi way triggering (sound, visual, vibration)

### **3. Task Creation and Accessibility Customization**

Enables users to create tasks with extended personalization such as priority tagging, visual formatting, and accessibility-related features.

* Decorator pattern supports runtime customization:  
  + Priority levels (e.g., High, Medium, Low)
  + Font size (small/medium/large)
  + High contrast or readable color schemes
* Accessibility preferences (like vibration-only or screen reader hints) are preserved per user profile
* All enhanced task configurations are reflected in the UI and persisted locally
* Ensures proper validation (e.g., required fields, visual conflicts)

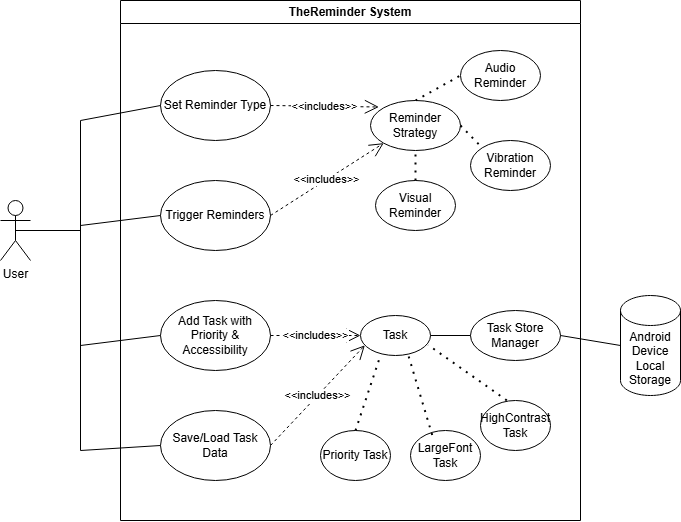
### **4. Local Task Storage Management**

Handles saving, loading, and deleting task data in a consistent and centralized manner using SQLite.

* All data operations are routed through a Singleton-based storage manager
* Guarantees task integrity and avoids redundant instances
* Reflects updates and deletions in the task list and scheduled reminders

# 

# Use Case Diagram



### **Use Case 1: Set Reminder Type**

**Description:** Allows users to choose how they want to be reminded (audio, visual, vibration) when a task is due.

**Actors:** User

**Preconditions:** User is creating or editing a task

**Main Flow:**

User selects a task or starts creating a new one

Chooses one or more reminder types (audio, vibration, visual)

Saves the task with selected reminder configuration

**Postconditions:** The selected reminder strategy is applied to the task and will execute when the task is due

**Exceptions:**

* No reminder type selected → A default type is applied or user is prompted
* Selected type not supported on device (e.g., no vibration motor) → UI disables that option or shows a message

**Applied Pattern:** **Strategy Pattern** – Allows flexible switching between different reminder behaviors without modifying core task logic.

### 

### **Use Case 2: Trigger Reminders**

**Description:** Handles the moment a reminder needs to go off and informs the user through chosen channels.

**Actors:** System (background service)

**Preconditions:** A task with a valid due time and reminder type is saved

**Main Flow:**

System checks task deadlines periodically

When task time matches current time, triggers the reminder logic

Notifies all relevant handlers (e.g., play sound, vibrate, show visual alert)

**Postconditions:** Reminder notification is executed using the selected method(s)

**Exceptions:**

* Background service is killed → Task is missed; may require retry mechanism
* Reminder handler fails (e.g., audio not playing) → Error is logged; fallback mechanism used

**Applied Pattern:** **Observer Pattern** – ReminderClock notifies multiple observers like audio, visual, or haptic handlers.

### 

### **Use Case 3: Add Task with Priority & Accessibility**

**Description:** Enables users to create tasks and apply additional behaviors like priority tagging or accessibility-based formatting.

**Actors:** User

**Preconditions:** User is on the task creation screen

**Main Flow:**

User clicks on "Add Task"

Enters task title and details

Applies settings like high priority or large font

Saves the task

**Postconditions:** A task is saved with optional enhancements that affect its visual or behavioral output

**Exceptions:**

* Priority level not selected → Default priority is used
* UI decorators conflict (e.g., large font + long title) → Layout adjusts automatically or warns the user

**Applied Pattern:** **Decorator Pattern** – Dynamically adds new behaviors (like visual emphasis or accessibility formatting) to the base Task object.

### 

### **Use Case 4: Save/Load Task Data**

**Description:** Handles saving task data locally and retrieving it reliably across the app lifecycle.

**Actors:** System

**Preconditions:** User performs an action that requires storing or loading tasks

**Main Flow:**

User creates, edits, or deletes a task

System interacts with local SQLite database via a single access point

Task data is stored or retrieved as needed

**Postconditions:** Task list is updated based on the latest database state

**Exceptions:**

* SQLite access fails (e.g., permissions, corruption) → App shows error and halts operation gracefully
* Singleton instance not initialized → App logs error and attempts to reinitialize

**Applied Pattern:** **Singleton Pattern** – Ensures consistent, single-instance access to local storage layer across all app components.

# 7. Non-Functional Requirements

**Usability**:  
 The app should be straightforward and easy to navigate for all users, including those with impairments. It should be easy to understand and use, with various accessibility features.

**Performance**:  
Primary features like task creation, update and deletion should be done within 2 seconds. The reminders should trigger in time.

**Portability**:  
 The app will only work on android devices.

**Reliability**:  
 The system should be stable and have minimal crashes and bugs.

# 8. External Interface Requirements

**Flutter Framework & Plugins**:  
 TheReminder is built using Flutter and depends on external packages such as flutter\_local\_notifications and vibration to handle reminder functionalities across devices.

**Android System Services**:  
 The app interacts with Android system services for notification delivery and haptic feedback. These interactions are handled through the standard Flutter plugin ecosystem and do not require custom system-level integration.

# 9. User Interfaces

#### **1. Task List Screen**

**Purpose**: Displays the list of current tasks and provides basic task interaction options.

**Features**:

* View all tasks, filtered by status (pending/completed)
* Tap a task to view or edit its details
* Swipe to delete or mark as done
* Filter or sort tasks by priority or category

#### **2. Add/Edit Task Screen**

**Purpose**: Enables users to create a new task or modify an existing one.

**Features**:

* Input fields: title, description, due date/time
* Select reminder types: audio, vibration, visual
* Choose category and priority level
* Save or cancel task changes

#### **3. Settings Screen**

**Purpose**: Central hub for both general and accessibility-related preferences.

**Features**:

**a) General Settings**

* Clear all completed tasks
* Reset all saved user data
* View help, about, and application version info

**b) Accessibility Settings**

* Toggle high contrast mode for better visual clarity
* Adjust font size (e.g., small, medium, large)
* Enable or disable screen reader hints
* Set default reminder mode (e.g., vibration-only or audio + visual)

#### **4. Reminder Notification Pop-ups**

**Purpose**: Notify users of upcoming tasks through their selected reminder method.

**Features**:

* Show pop-up reminders based on due time and selected type
* Quick actions (e.g., mark task as complete)
* Respect user accessibility settings (e.g., haptic feedback, sound cues, or screen reader-friendly labels)

# 10. Software Interfaces

TheReminder application interacts with several third-party libraries and system-level services to deliver its core functionalities. These interfaces enable the app to manage local data, schedule reminders, and support accessibility features without requiring external servers.

#### **1. Flutter SDK**

* The app is built using the Flutter framework, which provides the foundational UI and platform abstraction layers.
* Dart is used as the primary programming language for logic and UI development.

#### **2. SQLite (Local Database)**

* Used to store task data, reminder settings, user preferences, and accessibility configurations.
* The Flutter plugin sqflite is used to interface with SQLite from the Dart environment.

#### **3. flutter\_local\_notifications**

* A Flutter plugin used to schedule and display local notifications on Android.
* Supports audio and visual alert configurations.
* Interacts with Android’s native notification system.

#### **4. vibration**

* A Flutter plugin that enables triggering the device's haptic feedback for vibration-based reminders.
* Interfaces with Android’s hardware vibration API.

#### **5. flutter\_screen\_reader / accessibility\_plugins (if used)**

* Optional packages to enhance accessibility features like screen reader support or large text toggles.

# **Task Matrix**

| **Task** | **Responsible** |
| --- | --- |
| **Product Perspective** | **Özge Doğan** |
| **Product Functions** | **Beyzanur Zeybek** |
| **User Characteristics** | **Beyzanur Zeybek** |
| **Constraints** | **Özge Doğan** |
| **System Features & Use Cases** | **Talha Akbulut** |
| **Use Case Diagram** | **Talha Akbulut, Özge Doğan and Beyzanur Zeybek** |
| **Non-Functional Requirements** | **Beyzanur Zeybek** |
| **External Interface Requirements** | **Özge Doğan** |
| **User Interfaces** | **Talha Akbulut** |
| **Software Interfaces** | **Talha Akbulut** |

# 