SOFTWARE DESIGN DOCUMENT
for
Smart Personal Task Manager

V1.1

13 December 2025

Prepared by

Ahmet Harun Çakır

Beyzanur Yılmaz

Büşra Sarı

Selin Duru Derindağ

Soner Güneş

Sümeyye Zeynep Gürbüz

TABLE OF CONTENTS

1 Introduction

1.1 Purpose of the system

The purpose of the Smart Personal Task Manager (SPTM) is to provide an intelligent, cross-platform system that aligns daily task management with long-term personal mission statements. Unlike standard to-do lists, SPTM integrates structured prioritization frameworks—specifically Stephen Covey's Time Management Matrix and David Allen's GTD—to ensure user activities support meaningful personal growth.

1.2 Design goals

The design of the SPTM prioritizes the following technical and quality goals based on the non-functional requirements:

- Offline-First Availability: The system must function fully without an internet connection, synchronizing data only when connectivity is restored.
- Cross-Platform Consistency: The architecture must support a unified experience across Web and Mobile (Android/iOS) clients using shared business logic where possible.
- Responsiveness: User interactions must render within 1 second, with data loading under 2 seconds.
- Modularity: The system must be decomposable into distinct subsystems (Mission, Task, Calendar) to allow independent updates and future extensibility.
- Data Integrity: Synchronization conflicts (e.g., concurrent edits on different devices) must be resolved deterministically to prevent data loss.0

1.3 Definitions, acronyms, and abbreviations

- **SPTM:** Smart Personal Task Manager
- **GTD:** Getting Things Done (Productivity Methodology)
- **DTO:** Data Transfer Object
- **API:** Application Programming Interface
- **JWT:** JSON Web Token (for stateless authentication)
- **Covey Matrix:** A 2x2 prioritization grid (Urgent/Important)
- *See Section 5 for the complete Glossary.*

1.4 References

- Requirements Analysis Document (RAD) for Smart Personal Task Manager.
- Stephen R. Covey, The 7 Habits of Highly Effective People.
- David Allen, Getting Things Done.

1.5 Overview

This document details the architectural design of SPTM. Section 2 analyzes the current fragmented landscape. Section 3 proposes a layered client-server architecture with offline capabilities. Section 4 defines the specific services provided by each subsystem.

2 Current software architecture

2.1 Analysis of Current Systems

As noted in the requirements analysis, there is currently **no single integrated system** that connects daily tasks to long-term mission statements. Instead, the "current architecture" utilized by users consists of a fragmented collection of independent tools:

2.2 Architectural Issues to Address

The proposed system must resolve specific architectural deficiencies in the current "manual integration" approach:

1. **Data Silos:** Mission data and Task data exist in separate formats that cannot interact.
2. **Lack of Feedback Loops:** Progress tracking is manual and often neglected.
3. **Context Switching:** Users must switch applications to view their schedule vs. their tasks, leading to cognitive overhead.

3 Proposed software architecture

3.1 Overview

The SPTM will utilize a **Layered Client-Server Architecture** with a "Thick Client" approach to support offline functionality.

- **Presentation Layer (Client):** Mobile and Web applications that handle UI rendering and local logic.
- **Application Logic Layer (API):** A RESTful API handling synchronization, complex analytics, and third-party integrations.
- **Data Layer:** Local databases (SQLite/Realm) on clients for offline storage, and a central relational database (PostgreSQL) in the cloud for synchronization.

3.2 Subsystem decomposition

The system is decomposed into five logical subsystems based on the high-level components defined in the RAD.

1. Mission Management Subsystem

Responsible for the "Strategic" layer of the application.

- Responsibilities: Manages the lifecycle of MissionStatement and SubMission entities. Handles version history of missions.

Key Classes: MissionStatement, SubMission, MissionVersion.

2. Task Management Subsystem

Responsible for the "Tactical" layer of the application.

- Responsibilities: Manages Task CRUD operations, hierarchical parent/subtask relationships, and Context tagging. Implements the CoveyQuadrant logic to classify tasks by urgency/importance.

Key Classes: Task, TaskHierarchy, CoveyQuadrant, TaskContext.

3. Calendar Integration Subsystem

- Responsibilities: Interfaces with external providers (Google/Apple). Converts CalendarEvent objects into Task objects and handles bidirectional synchronization.

  This system follows the **“adapter” design pattern**.

Key Classes: CalendarSync, CalendarEvent.

4. Progress & Analytics Subsystem

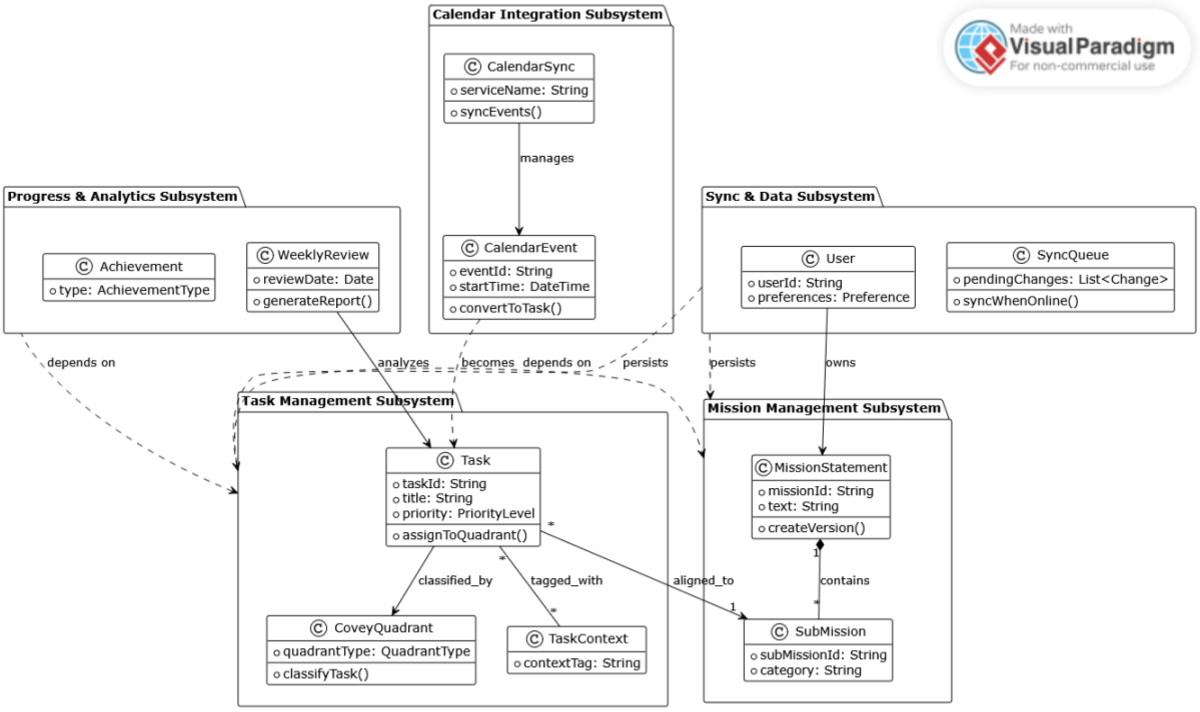
- Responsibilities: Aggregates data to generate WeeklyReview reports and track achievement milestones. Calculates time allocation across mission areas.

Key Classes: ProgressTracker, WeeklyReview, Achievement.

5. Data Synchronization Subsystem

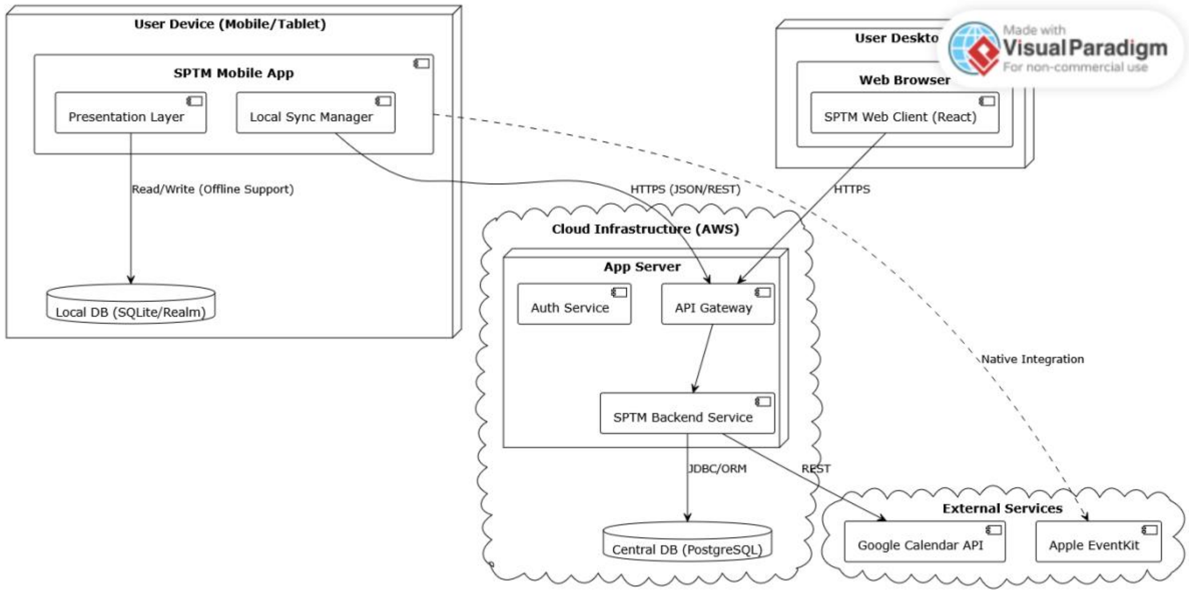
- Responsibilities: Manages the SyncQueue. Detects network connectivity changes, pushes PendingChange objects to the server, and resolves conflicts.

Key Classes: SyncQueue, PendingChange, DataExport.



3.3 Hardware/software mapping

Component	Software/Technology Stack	Hardware Mapping
Mobile Client	Flutter (Cross-platform)	User's Smartphone (iOS/Android)
Web Client	React.js	User's Laptop/Desktop Browser
Local Storage	SQLite (Encrypted)	User's Device Storage
Backend API	SpringBoot (RestAPI)	Cloud Server Instance (e.g., AWS EC2)
Central Database	PostgreSQL	Managed Cloud Database (e.g., AWS RDS)
External Services	Google Calendar API, Apple EventKit	Third-party Cloud Infrastructure



3.4 Persistent data management

The system requires a Relational Database Management System (RDBMS) to maintain the strict referential integrity between Missions, Tasks, and Users.

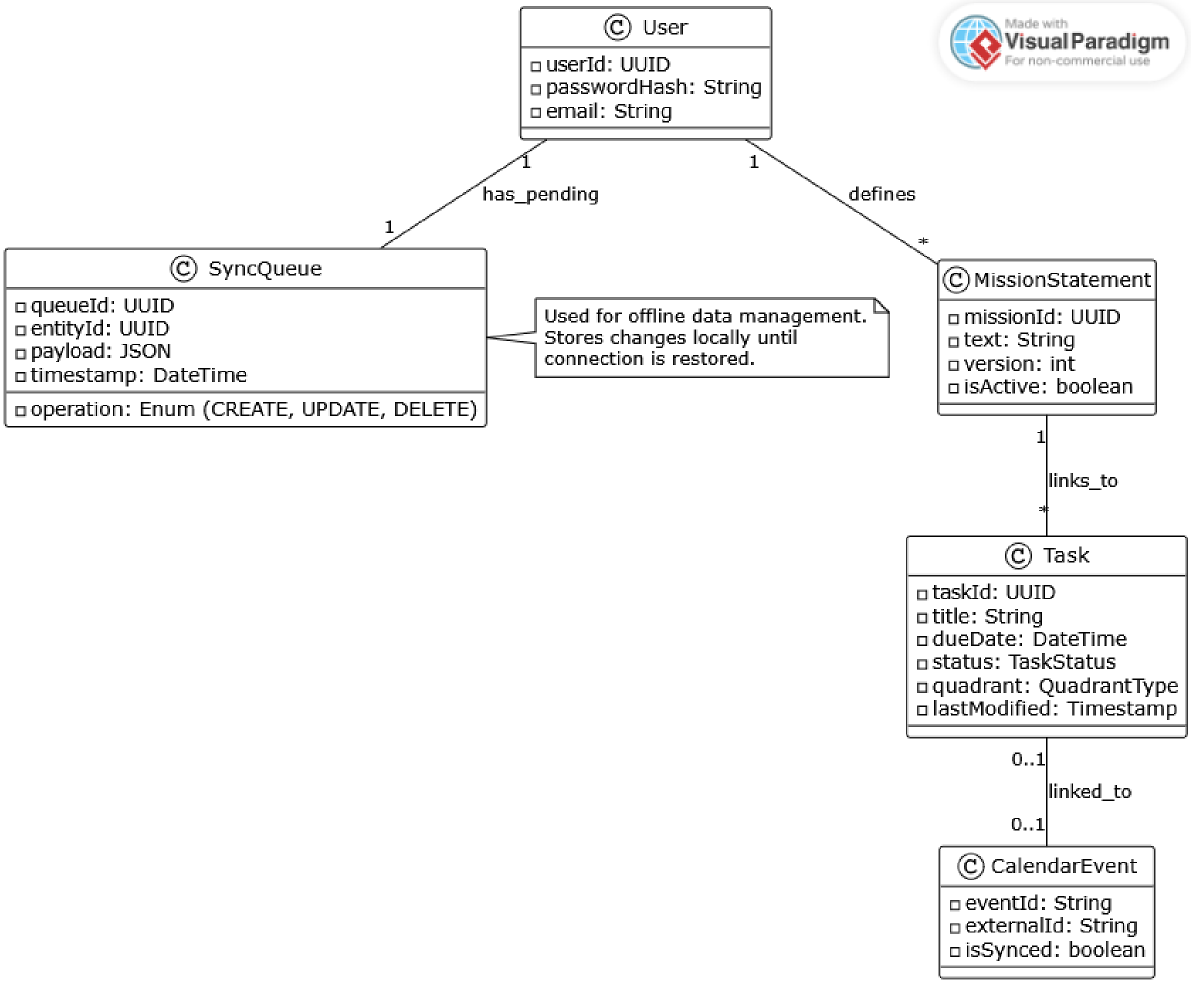
Key Entity Relationships:

- **One-to-Many:** A User has many MissionStatement versions.
- **One-to-Many:** A MissionStatement has many SubMission categories.
- **Many-to-One:** A Task belongs to one SubMission.
- **Many-to-Many:** A Task can have multiple TaskContext tags.

Data Storage Strategy:

1. **Local:** Stores a full replica of the user's data to ensure FR-24 (Offline Access) is met.

2. **Cloud:** Acts as the source of truth. The SyncQueue table tracks the lastAttempt and retryCount for data consistency.



3.5 Access control and security

Authentication:

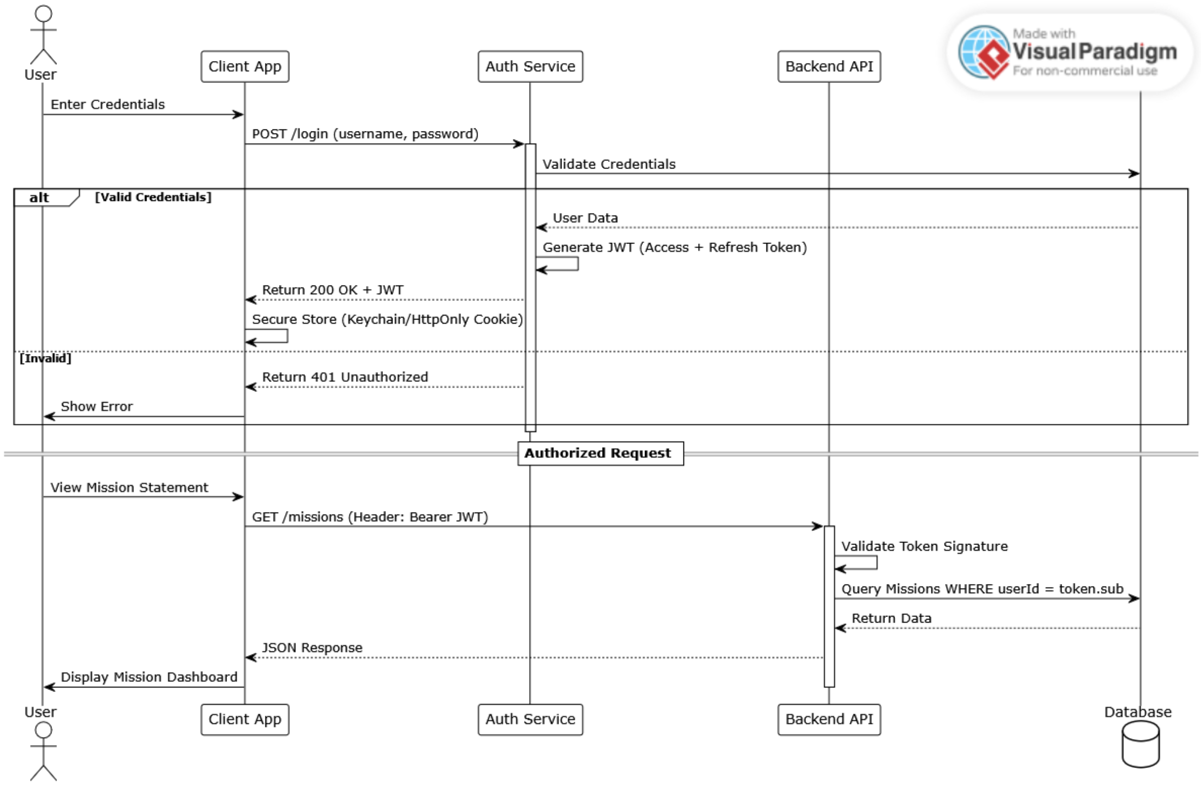
- The system will use **JWT (JSON Web Tokens)** for stateless authentication between the client and server.

Encryption:

- **At Rest:** Local databases on mobile devices must be encrypted to satisfy FR-25.
- **In Transit:** All synchronization traffic uses HTTPS (TLS 1.2+).

Access Matrix:

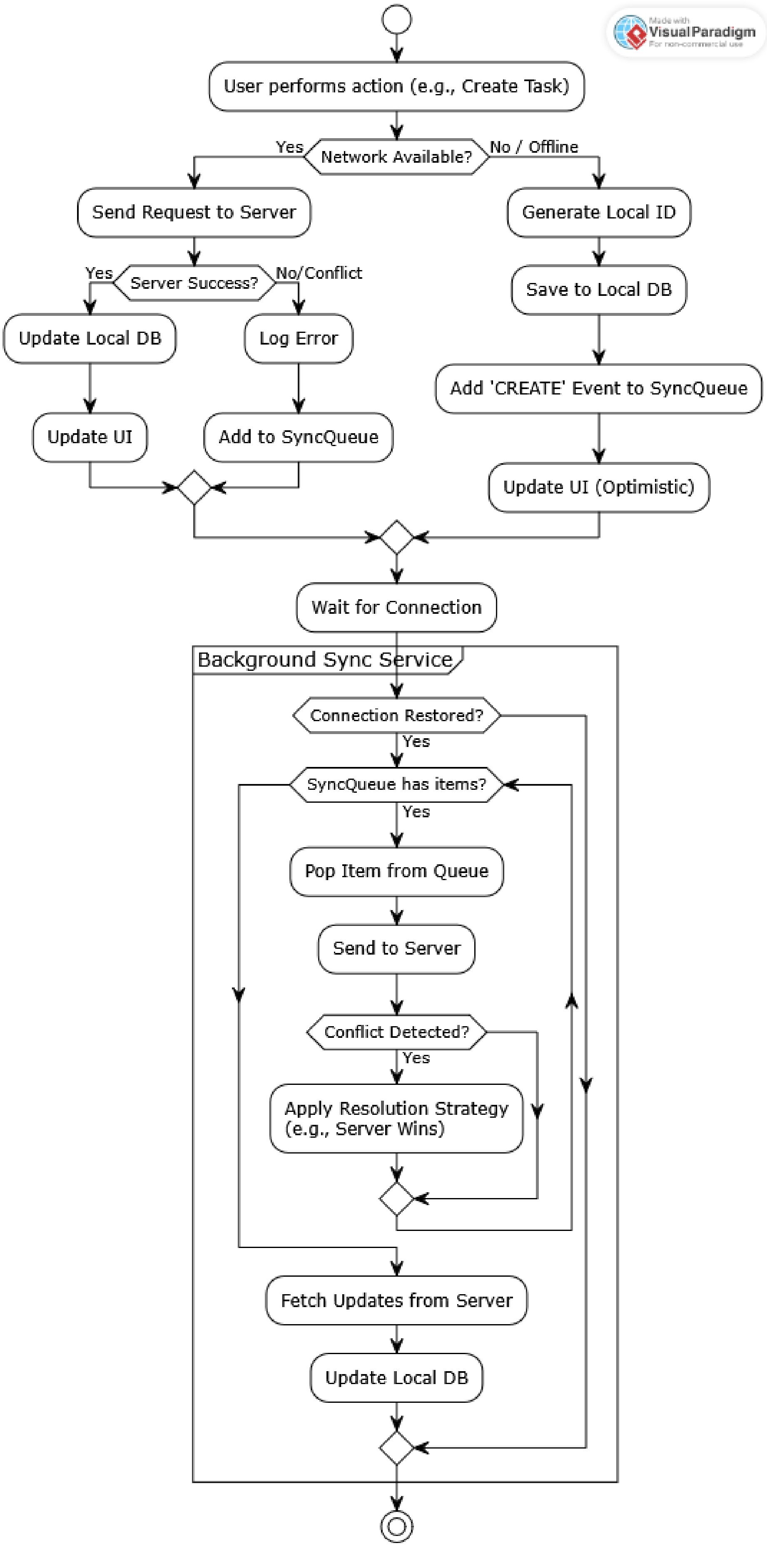
- The system is single-user focused per account. Users have full CRUD access only to their own data (row-level security based on userId).



3.6 Global software control

The system follows an **Event-Driven Control Flow** on the client side:

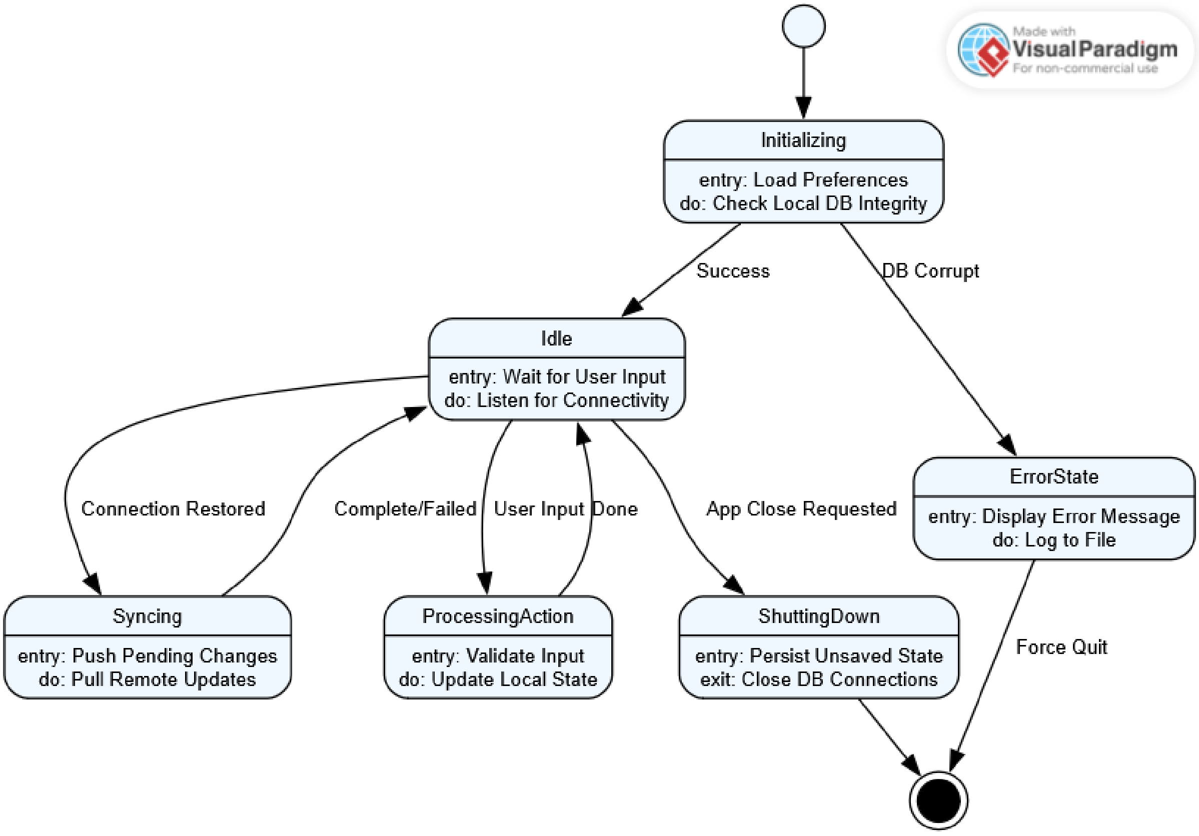
- **User Action:** User creates a task -> UI updates immediately (Optimistic UI) -> Event added to SyncQueue.
- **System Event:** Network connection restored -> SyncManager triggers syncWhenOnline().
- **External Event:** Webhook/Polling from Google Calendar detects new event -> Notification triggered.


User performs action (e.g., Create Task)
Network Available?
Yes
No / Offline
Send Request to Server
Server Success?
Yes
No/Conflict
Update Local DB
Log Error
Update UI
Add to SyncQueue
Generate Local ID
Save to Local DB
Add 'CREATE' Event to SyncQueue
Update UI (Optimistic)
Wait for Connection
Background Sync Service
Connection Restored?
Yes
SyncQueue has items?
Yes
Pop Item from Queue
Send to Server
Conflict Detected?
Yes
Apply Resolution Strategy (e.g., Server Wins)
Fetch Updates from Server
Update Local DB
Made with Visual Paradigm For non-commercial use

3.7 Boundary conditions

Startup: The app initializes the NotificationManager and checks SyncQueue for pending uploads. It loads cached data immediately for performance.
Shutdown: Ensures any in-progress database writes are committed to local storage to prevent corruption.

Error Behavior (Network): If synchronization fails, the system increments the retryCount in SyncQueue and schedules a retry, notifying the user only if manual intervention is required.



4 Subsystem services

This section defines the public interfaces for the subsystems identified in Section 3.2, derived from the Object Models in the RAD.

4.1 Mission Management Services

Operation	**Description**
createPersonalMission(userId, text)	Creates the initial mission statement[40].
createVersion(missionId)	Archives current mission and creates a new editable version[41].

Operation	Description
addSubMission(missionId, category)	Adds a goal category (e.g., "Health") to a mission[42].
getMissionHierarchy(userId)	Returns the tree of Mission -> SubMissions.

4.2 Task Management Services

Operation	Description
createTask(taskDTO)	Creates a new task with title, due date, and priority[43].
assignToQuadrant(taskId)	Calculates urgency/importance and assigns Covey Quadrant[44].
addContext(taskId, contextTag)	Adds GTD tags (e.g., @home) to a task[45].
getTasksByFilter(filterCriteria)	Returns tasks filtered by mission, tag, or status[46].

4.3 Calendar Services

Operation	Description
syncWithCalendar(serviceName)	Initiates auth flow and syncs events[47].
convertEventToTask(eventId)	Creates a Task object from a CalendarEvent[48].
resolveConflict(strategy)	Handles data clashes based on settings (e.g., SERVER_WINS)[49].

4.4 Analytics Services

Operation	Description
conductReview(userId)	Starts the Weekly Review workflow[50].
generateVisualReport(period)	Returns data for completion trends and time allocation[51].
recordAchievement(type, missionId)	Logs milestones like "Streak" or "Goal Completion"[52].

5 Glossary of Terms

Term	Definition
SPTM	Smart Personal Task Manager (the system described in this document)[53].
Mission Statement	A written statement expressing core values, serving as the root for all tasks[54].
Covey Matrix	A prioritization model dividing tasks into four quadrants based on Urgency and Importance[55].

Term	Definition
SyncQueue	A local data structure that holds changes (Create/Update/Delete) while offline, waiting for synchronization[56].
Sub-mission	A specific goal category derived from the main mission (e.g., "Career")[57].
Weekly Review	A periodic workflow where users reflect on progress and plan the upcoming week[58].

6 Traceability

The following matrix traces the Subsystem Design to the Functional Requirements (FR) defined in the RAD .

Subsystem	Functional Requirements Addressed
Mission Management	**FR-1, FR-2, FR-3, FR-4, FR-5, FR-6** (Creation, Editing, Versioning, Hierarchy)
Task Management	**FR-7, FR-8, FR-9, FR-10, FR-11, FR-13, FR-12** (CRUD, Hierarchy, Covey Matrix, GTD Tags)
Calendar Integration	**FR-14, FR-15, FR-16, FR-17** (Sync, View, Convert, Notify)
Progress & Analytics	**FR-18, FR-19, FR-20, FR-21** (Tracking, History, Reports, Review Prompts)
Data & Sync	**FR-22, FR-23, FR-24, FR-25** (Cross-platform, Offline Access, Security)

This design ensures all requirements, particularly the high-priority goal of linking tasks to mission statements (FR-5, FR-7), are structurally supported by the architecture.