Detailed Design Document

Introduction

The Personal Time Management System is designed to help users plan, track, and analyze their time usage through an integrated web application that combines schedule planning with real-time tracking capabilities. This document outlines the design of the web-based time management system, detailing its architecture, component interactions, and technical specifications.

System Overview

System Objectives

- Enable users to schedule events with flexibility (fixed, flexible, fluid).
- Provide real-time activity tracking integrated with scheduled events.
- Offer advanced data visualization and analysis for time management insights.

Components

- User Interface (UI): Developed using React, this provides scheduling, tracking, and analytics visualization.
- Backend: Built with Node.js, it manages API requests, event scheduling logic, and database interactions.
- Database: Stores user data, events, time logs, and analytics.

Design Considerations

Assumption

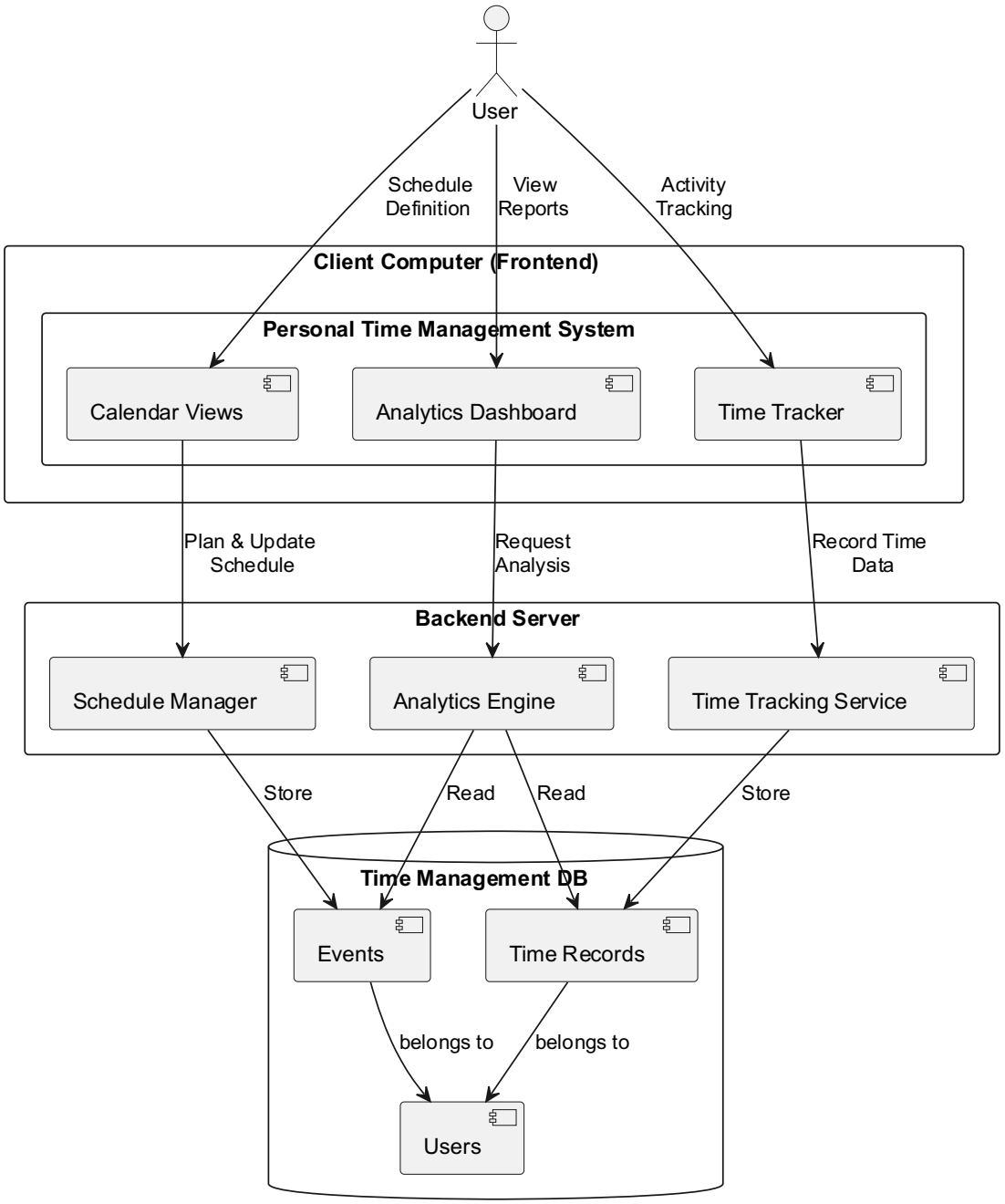
- The project is a proof of concept rather than a commercial-grade product.
- Users will interact with the system primarily on desktop web browsers.
- Modern web browser support (Chrome, Firefox, Safari)
- Maximum of 1000 concurrent users

Constraints

- Browser-based implementation only (no native mobile app)
- Limited to foundational features and functionality for demonstration purposes.

System Architecture

Architecture Diagram



Frontend Layer (Client Computer)

The frontend layer hosts the Personal Time Management System, which contains three primary user interface components:

Calendar Views

The Calendar Views component serves as the primary scheduling interface, providing:

- Weekly calendar visualization with support for three event types (fixed, flexible, and fluid)
- Daily detailed view for schedule management
- Drag-and-drop interface for event manipulation
- Visual distinction between different event types through color coding
- Conflict detection warnings during event scheduling
- Integration with the Schedule Manager service for persistent storage

Time Tracker

The Time Tracker provides real-time activity monitoring features:

- Start/stop controls for activity tracking
- Current activity display with time
- Manual time entry for missed activities
- Chronological list view of activities
- Automatic synchronization with Time Tracking Service

Analytics Dashboard

The Analytics Dashboard delivers insights about time usage through:

- Interactive visualizations comparing planned versus actual time usage
- Customizable reporting periods (daily, weekly, monthly)
- Time distribution analysis by category and activity type
- Real-time data updates from the Analytics Engine

Backend Layer (Backend Server)

The backend layer consists of three specialized services that process and manage all system data:

Schedule Manager

The Schedule Manager implements intelligent scheduling logic:

- Forward Checking algorithm for efficient event placement
- Backtracking capabilities for resolving scheduling conflicts
- Schedule optimization for flexible and fluid events
- Direct database interaction

Time Tracking Service

The Time Tracking Service manages activity monitoring:

- Real-time activity state management
- Automatic time record generation
- Synchronization with planned events
- Historical record maintenance
- Direct database interaction for time record storage

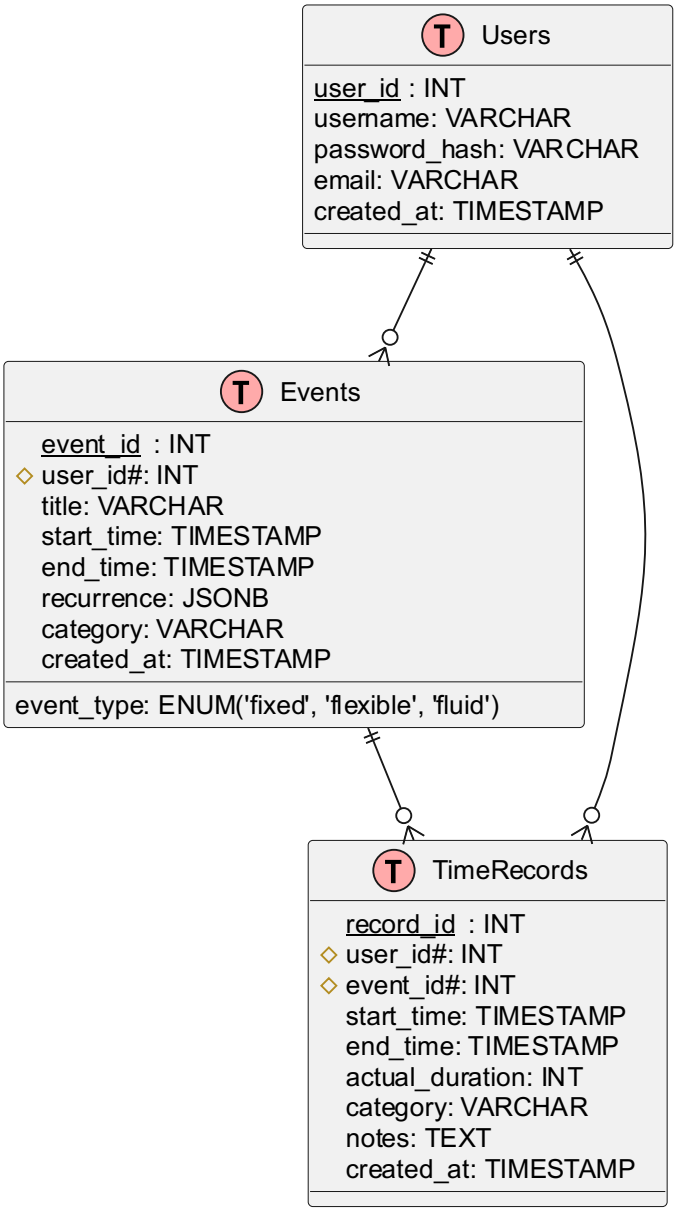
Analytics Engine

The Analytics Engine processes time data to generate insights:

- Implementation of Dynamic Time Warping algorithm for pattern matching
- Calculation of efficiency metrics and time utilization rates
- Historical trend analysis

Data Design

Database Schema



The database layer maintains three interconnected data stores:

Events Table

Stores all schedule-related data:

- Event definitions (fixed, flexible, fluid)
- Scheduling constraints and parameters
- Recurrence patterns
- Foreign key relationship to Users table

Time Records

Maintains actual time usage data:

- Activity tracking records
- Start and end timestamps
- Category and description information
- Foreign key relationship to Users table

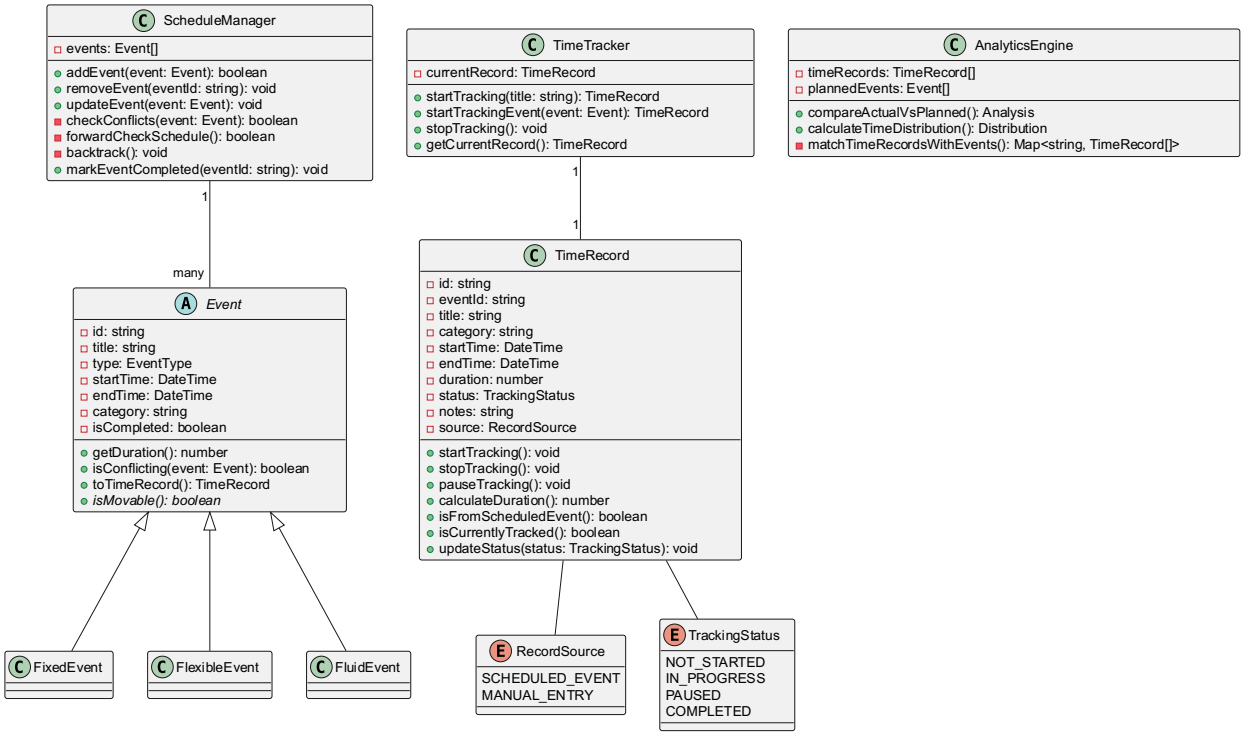
Users

Central user information store:

- User authentication data
- Profile information
- Preferences and settings
- Primary key referenced by Events and Time Records

Class Design

Class Diagram



User Interface Design

Weekly View

The weekly view serves as the primary planning interface, displaying a traditional calendar grid with seven columns representing days of the week. Time slots run vertically, allowing users to visualize their schedule across the week. Each event appears as a block within the grid, with visual distinctions between categories. The interface supports drag-and-drop functionality for event management and includes a simple event creation flow through both a dedicated button and direct grid interaction.

Daily View

The daily view provides a detailed timeline of a single day's schedule. This view emphasizes time slots and event details, displaying more information about each scheduled item. Users can interact directly with the timeline to create or modify events. The interface maintains consistency with the weekly view while offering finer granularity for daily planning.

Time Tracking Interface

The time tracking interface adopts a minimalist design focused on the current activity. A prominent timer control allows users to start and stop activity tracking. The interface displays both the active timer and a chronological list of completed activities. Integration with the planned schedule allows quick selection of scheduled events for tracking, while also supporting manual entry for unplanned activities.

Analytics Dashboard

The analytics interface presents time usage data through a series of visualizations. The layout uses a modular design where each analysis component (time distribution, planned vs. actual comparison, trends) occupies its own section. Interactive elements allow users to adjust time ranges and drill down into specific metrics. The interface prioritizes clarity in data presentation while maintaining interactivity for exploration.

Common Elements

Each interface shares consistent navigation and interaction patterns. The system uses a standard header for navigation between views and maintains unified styling for interactive elements like buttons and forms. Event creation and editing flows remain consistent across all views where they appear.