Notifications Center Documentation

# Introduction

The Notifications Center is a system that allows users to receive, view, and manage notifications about important account activity, performance updates, and system messages. Notifications are displayed in a centralized panel, accessible via a notification bell icon. Users are alerted in real-time about significant activities, such as critical system updates, account changes, or performance metrics.

# Key Features

1. Unread Notifications Badge: Unread notifications are displayed as a badge on the notification bell icon, providing users with a quick visual cue.  
2. Dropdown for Recent Notifications: When users click the notification bell, a dropdown list is displayed, showing the most recent notifications.  
3. Mark Notifications as Read: Users can mark notifications as read when interacting with them, removing the unread indicator.  
4. Real-Time Notifications: Notifications are pushed to the client in real-time using WebSockets or Server-Sent Events (SSE).  
5. Distinct Colors & Icons: Notifications have different colors and icons based on their type (e.g., blue for updates, red for critical alerts, exclamation mark for errors).  
6. Timestamps: Each notification includes a clear timestamp indicating when it was triggered.  
7. Customizable Alerts: Users can manage which types of notifications they wish to receive, such as alerts, updates, or general system information.

# System Design

## 1. Frontend Design (React)

Notification Bell Icon:  
- A notification bell icon is displayed in the header or navigation bar of the user interface.  
- When there are unread notifications, a badge is shown on the bell icon.  
- Clicking on the bell icon opens a dropdown list of recent notifications.

Notification Dropdown:  
- The dropdown lists recent notifications, displaying the notification type, message, and timestamp.  
- Notifications are styled using distinct colors based on type (e.g., blue for updates, red for alerts).  
- Each notification includes an icon representing the type of event, such as:  
 - Exclamation mark for critical alerts.  
 - Check mark for success messages.  
 - Information icon for system updates.

Notification List:  
- The notification list should be scrollable if there are multiple items.  
- Users can interact with notifications, either dismissing them or navigating to relevant content.

## 2. Real-Time Notifications

WebSockets (Preferred) / Server-Sent Events (SSE):  
- WebSockets allow real-time, two-way communication between the client and server.  
- Server-Sent Events (SSE) offer simpler, one-way communication, ideal for sending real-time notifications to clients.

## 3. Notification Types and Styles

Types of Notifications:  
- Alert (Critical): Used for urgent messages, such as security alerts, password changes, or error notifications.  
- Update (Informational): Used for system or account updates, such as new features or settings changes.  
- Success: Positive messages such as account creation success or task completion.  
- Warning: Non-critical issues that need the user’s attention, such as low disk space or app issues.

Styles and Icons:  
- Notifications are color-coded for better clarity:  
 - Red for Alerts (e.g., critical alerts like account changes).  
 - Blue for Updates (e.g., system updates or feature announcements).  
 - Green for Success (e.g., task completed successfully).  
 - Yellow for Warnings (e.g., issues requiring attention but not critical).  
- Icons represent each type of notification:  
 - Exclamation mark for critical alerts.  
 - Information icon for updates.  
 - Checkmark for success.  
 - Warning triangle for warnings.

## 4. Backend Design

To manage notifications, the backend is responsible for sending notifications to clients in real-time. The notifications are stored in the database, and their status (read/unread) is updated based on user interaction.

## 5. Toast Notifications (Optional)

For immediate, non-intrusive alerts, you can use toast notifications that appear briefly on the screen. These are suitable for providing feedback or showing success/error messages.

## 6. Data Flow

1. User Activity Occurs:  
- A user triggers an event, such as account changes, critical system updates, or performance alerts.  
2. Notification Sent to Server:  
- The server detects the event and pushes a notification to the appropriate users.  
3. Client Receives Notification:  
- The client displays the notification in the dropdown, and a badge appears on the bell icon if the notification is unread.  
4. User Interacts with Notifications:  
- The user clicks on a notification to read it, marking it as read.  
5. Persist Notification Status:  
- The server updates the notification's read status in the database, ensuring synchronization across all devices.

## 7. Future Enhancements

1. Push Notifications: Integrate push notifications for mobile and desktop notifications.  
2. User Preferences: Allow users to manage notification preferences for different types of alerts.  
3. Priority Notifications: Implement priority levels to highlight more urgent notifications above others.

A screenshot of a computer

Description automatically generated