Software Design Specification

Notify App

Group 6

Athul V 17

Gilu Vijayan 28

Gouri P 29

Kailas Unni 35

TABLE OF CONTENTS

1. Introduction
    1. Purpose
    2. Scope
    3. Intended Audience and Reading Suggestions
2. User Interface Design
    1. Overview
    2. Design Elements
3. Data Model Design
    1. Overview
    2. Entities
4. Application Architecture
    1. Overview
    2. Presentation Layer
    3. Business Layer
    4. Datalink Layer
5. Implementation Details
    1. Technologies Used
    2. Development Environment
6. Conclusion
7. References

INTRODUCTION

Purpose

The purpose of a notification system that helps prioritize academic-related notifications is to streamline and manage the flow of information to students in a more effective way. With so much information coming from various sources such as official sites, faculties, and class representatives, it can be overwhelming and time-consuming to sift through everything to find what is most relevant and urgent. By managing academic-related notifications in a centralized system, students can also avoid missing important deadlines or events that may impact their academic progress. Additionally, the system can provide a more personalized and tailored experience by filtering notifications based on the student's course load, interests, and academic standing. Overall, a notification system that helps prioritize academic-related notifications can help students stay on top of their academic responsibilities and ensure they are getting the information they need to succeed.

Scope

The product scope for a notification application for students that helps prioritize notifications would include features and functionalities that improve the flow of academic-related information, streamline communication between students and academic staff, and provide a personalized and user-friendly experience. 1. Notification Management System: The application should have a robust notification management system that can effectively manage the flow of notifications from various sources such as official sites, faculties, and class representatives. 2. Notification Prioritization: The application should prioritize academic-related notifications such as upcoming exams, assignment deadlines, and important announcements over general notifications and event invitations. 3. Personalization: The application should provide a personalized experience by filtering notifications based on the student's course load, interests, and academic standing. 4. Security and Privacy: The application should include security and privacy requirements to protect students' personal information and academic data. 5. User Interface Design: The application should have a user-friendly interface that is easy to navigate for students, professors, and academic staff

Intended Audience and Reading Suggestions

The intended audience for a software requirements specification (SRS) document for a notification application for students would include project managers, software developers, quality assurance teams, system architects, and other technical stakeholders who are involved in the development and delivery of the software product. In addition, the intended audience for this particular SRS document would also include students, professors, and other academic staff who will be using the notification application. The document consists of a detailed description of all the functional and non-functional requirements and various perspectives of the product. It has all the features that are to be implemented in the product

USER INTERFACE DESIGN

Overview

The user interface design for the notify app aimed at helping students prioritize and organize notifications from faculty, representatives, and official sites should be intuitive, visually appealing, and user-friendly. Here's an overview of the key components and principles that should be considered for the UI design:

Dashboard/Home Screen: The dashboard or home screen should provide an overview of all the notifications in a visually organized and easy-to-understand manner. It should display notifications from different sources such as faculty, representatives, and official sites in a prioritized and categorized manner, allowing students to quickly identify and access important notifications.

Notification Cards: Each notification should be presented as a card with relevant information such as sender, subject, time, and importance level. The cards should be visually distinguishable with different colors or icons based on the source or priority of the notification, making it easy for students to differentiate between notifications and prioritize them accordingly.

Filtering and Sorting Options: The app should provide filtering and sorting options to allow students to customize how they view their notifications. This could include options to filter notifications by source, priority, date, or category, and sorting options such as sorting by importance, time, or sender. These features can help students easily find and focus on the most important notifications.

Notification Details: Clicking on a notification card should display the detailed information of the notification, including the message content, attachments, and any actions that need to be taken, such as replying or marking as read. Clear and prominent buttons or icons should be provided to allow students to take action on the notification easily.

Categories or Labels: The app can allow students to categorize or label notifications based on their preferences or priorities. This could include creating custom categories or using pre-defined labels such as "Important," "Urgent," "Academic," or "Social." This feature can help students further organize and prioritize notifications based on their own criteria.

Settings and Customization: The app should provide settings and customization options, such as allowing students to choose their preferred notification settings, such as sound, vibration, or pop-up alerts. Customization options could also include the ability to change the theme or layout of the app, allowing students to personalize their user experience.

Navigation and Backward Compatibility: The UI should be designed with easy and intuitive navigation, allowing students to quickly switch between different screens or sections of the app. Additionally, backward compatibility should be considered, ensuring that the app is accessible and usable across different devices, including smartphones, tablets, and desktop computers.

Help and Support: The app should provide help and support features, such as a tutorial, FAQs, or a contact information for technical support. This can assist students in getting familiar with the app and resolving any issues they may encounter.

Design elements

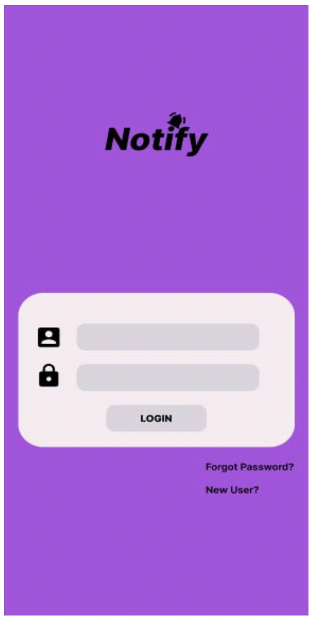
Login

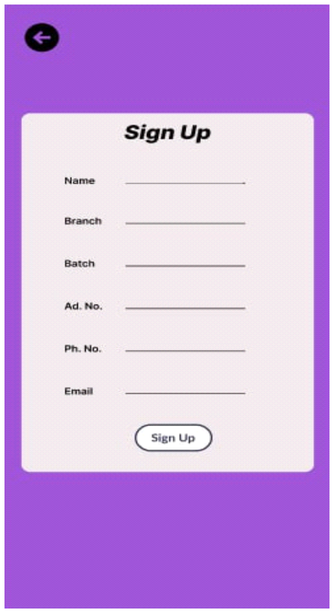
Users can login by typing their credentials.

• Users can select their role.

• 5 chances are given to enter password.

• Password recovery through email.



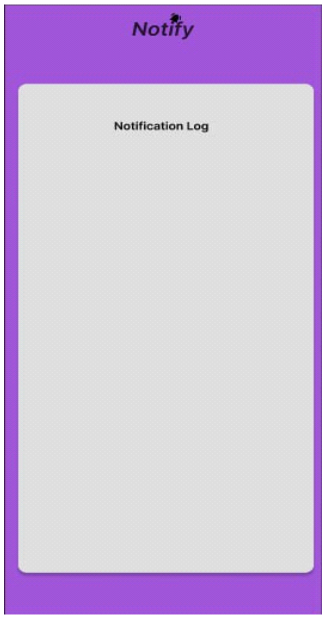


Notification log

This is the home page that comes after login. Users can view their notifications.

• Notifications are auto prioritized based on keywords.

• Users can filter their notifications in any manner including topic wise, priority, time, source, etc.

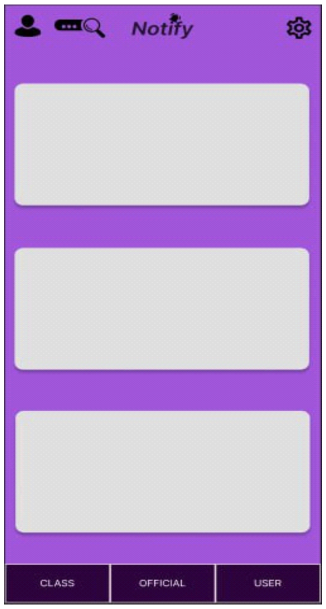


View Notifications

Users can select a particular notification and view in detail.

• Modify reminders.

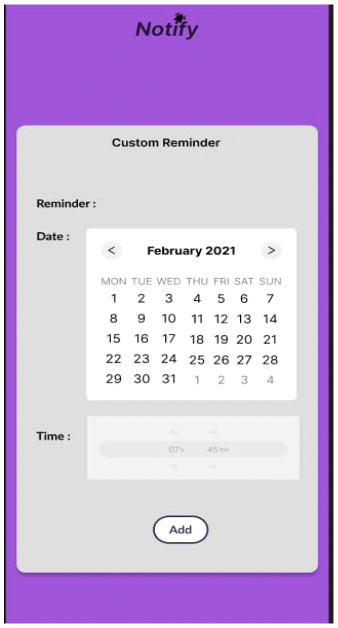
• Modify priority manually.



View reminders

Users can view their reminders and modify these accordingly.

• Modify already set reminders.



Notification History

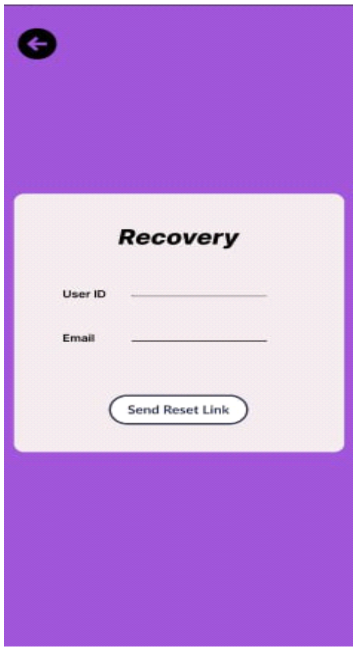
Users can view notification history.

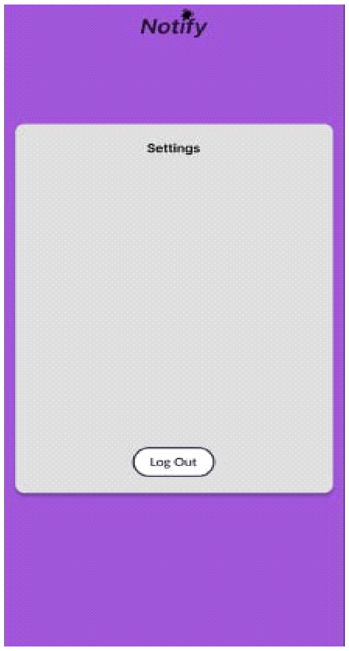
• These include previous notifications that are already expired.

Create custom reminders

Users are able to create custom reminders for their personal needs.

• These reminders can also be auto prioritised.





DATA MODEL DESIGN

Overview

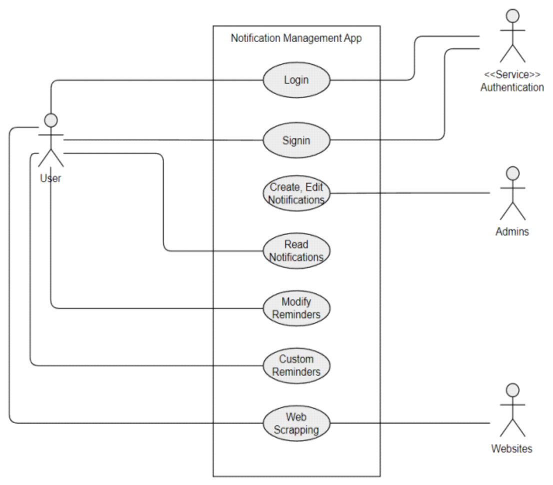
This data model allows the notification application to efficiently manage notifications and user interactions, while also enabling users to prioritize their notifications. By storing user interactions with notifications, the application can prioritize notifications based on each user's activity history and ensure that important notifications are not missed.

User: It is an entity that stores information about users, including their name, email, and login credentials. The users include admins, students, faculties and officials.

Notification: It is an entity that stores information about a notification, including the title, message, and sender.

There exit one to many relationship between user and notification. A user can have many user notifications.

USECASE DIAGRAM



DATABASE SCHEMA

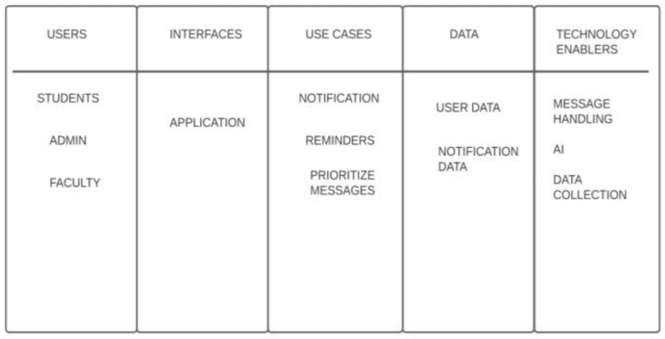
Users

username	String
role	Type:string Enum:[“USER”,”ADMIN”]
fullname	String
email	String
phone_no	String
password	String

Notifications

notification_id	String
message	String
priority	Integer
date	Date

ARCHITECTURAL OVERVIEW



USERS	INTERFACES	USE CASES	DATA	TECHNOLOGY ENABLERS
STUDENTS ADMIN FACULTY	APPLICATION	NOTIFICATION REMINDERS PRIORITIZE MESSAGES	USER DATA NOTIFICATION DATA	MESSAGE HANDLING AI DATA COLLECTION

The application will be designed using a three-tier architecture, consisting of the following layers:

- Presentation layer
- Business logic layer
- Data access layer

Presentation Layer:

The presentation layer is, as the name suggests, the portion that is shown to the users. The user interface and communication layer of the architecture is where users engage with application on the front end while the back end program gathers data and handles requests. Desktop apps may be written in a variety of languages, depending on the e-commerce platform, but here the presentation layer is frequently developed using flutter.

Business Logic Layer:

The business logic layer will contain the application logic and will be implemented using AI.

Data Access Layer:

The data access layer will be responsible for accessing the data in the database and will be implemented using google firebase.

IMPLEMENTATION OVERVIEW

Technologies used

The following technologies will be used to implement the application:

- Flutter for the user interface

  Flutter provides a fast, efficient, and customizable framework for mobile app development, making it a popular choice for mobile app developers.

- AI for the business logic layer

  AI to prioritize notifications in mobile apps can lead to a more personalized, engaging, and efficient user experience, while also reducing distractions and costs.

- Google firebase for the data access layer

  We have chosen Google Firebase as our backend, as it is capable of authentication, cloud Storage, notification

Development environment

The application will be developed using the following development tools:

- Visual Studio Code as the integrated development environment
- Git for version control

- GitHub for code repository hosting

Conclusion

This software design document provides a detailed overview of the Notify application design. It covers the user interface design, data model design, application architecture, and implementation details. This document will serve as a guide for the development team during the implementation phase of the project.

Reference