Text

Description automatically generated

Software Engineering Department

Ort Braude College

Capstone Project Phase A

**Notifications management system**

**Contents**

**0.Abstract -----------------------------------------------------------------------------------------------3**

**1. Introduction-------------------------------------------------------------------------------------------------3-4**

**2. Background--------------------------------------------------------------------------------------------------5-8**

**2.1 Existing solutions-----------------------------------------------------------------------------------------6**

**2.2. The Evolution of Notification Sender Systems ---------------------------------------------- 7-8**

**3. Expected Achievements ------------------------------------------------------------------------------ 8-10**

**3.1 Criteria for Success -------------------------------------------------------------------------------- 9-10**

**4. Engineering Process --------------------------------------------------------------------------------- 10-20**

**4.1 Processes--------------------------------------------------------------------------------------------10-12**

**4.1.1 Research and motivation ------------------------------------------------------------------ 10**

**4.1.2 Work process ------------------------------------------------------------------------------10-11**

**4.1.3 Constraints to be taken care of -------------------------------------------------------11-12**

**4.2 Product----------------------------------------------------------------------------------------------12-20**

**4.2.1 Requirements ------------------------------------------------------------------------------12-13**

**4.2.2 Diagrams ------------------------------------------------------------------------------------14-20**

**5. Testing-- -------------------------------------------------------------------------------------------------21-22**

**6. References -------------------------------------------------------------------------------------------------- 22**

**Abstract**

The notification management system described here empowers users to send notifications to targeted recipients without the need for handling or processing responses. It serves as a user-centric platform that enables individuals or groups to disseminate important information, updates, or alerts through various communication channels such as email, SMS, push notifications, or in-app messages. The system provides a user-friendly interface through which users can compose and customize notification messages. They can specify the desired timing or schedule for sending the notifications, select the intended recipients or recipient groups, and choose the appropriate delivery channels. The system ensures that notifications are sent according to the users' preferences and predefined schedules. Administrators or system operators play a role in configuring the system settings and granting users the necessary permissions to send notifications. They oversee the overall functionality and ensure the smooth operation of the system, but do not directly engage in handling the responses to the notifications. Recipients of the notifications receive the messages through their preferred communication channels. Although they may have the ability to respond to the notifications, the focus of this system is on enabling users to broadcast information rather than managing the ensuing interactions. By providing users with the capability to send notifications without the burden of response handling, the system streamlines communication processes, improves information dissemination, and empowers users to efficiently reach their target audience. It simplifies the sending of notifications, enhancing user engagement and facilitating effective communication within the specified scope of the system's capabilities.

# Introduction

With the abundance of information available to us, it's easy to miss important messages and updates. Notification management system s can help users stay on top of important information by delivering timely and relevant notifications to them. In some cases, users may not be aware of important events or updates relevant to them. Notification management system s can bridge this communication gap by delivering notifications directly to users, even if they are not actively using the system. In order to keep users engaged, it's important to provide them with timely and relevant updates. Notification management systems can help improve user engagement by delivering notifications that are tailored to the user's interests and preferences. To solve the problem, our notification management system will help users avoiding these problems. The system will help various stakeholders: (1) Business owners by helping them stay connected with their customers and keep them engaged with their products or services. Moreover, it can help businesses to improve their customer service by sending timely notifications regarding order updates, promotions, and other important information. (2) Marketing and sales teams by using it to send targeted, personalized notifications based on the user's behavior and interests to their customers, which will improve their conversion rates. (3) IT departments by using it to monitor and manage their systems and application purpose to detect and respond to issues quickly and efficiently, reducing downtime and improving performance. (4) Administrators by using it to communicate with their users or members. The system can help them send important announcements, updates, and reminders in a timely and organized manner. (5) End-users, which are the primary beneficiaries of a notification management system for receiving notifications. For example, if the system is a mobile app that sends push notifications, the users of the app benefit from receiving relevant and timely notifications.

The notification management system to in this work will include the following features:

Integration with other applications: The notification management system will be able to integrate with other applications and services that generate notifications, such as messaging apps, email clients, and social media platforms.Customization options: The notification management system will allow the sender to customize the content, format, and frequency of the notifications that are sent.Targeted delivery: The notification sender system will be able to deliver notifications to specific users or groups based on their preferences, location, and other criteria.Delivery channels: The notification management system will support multiple delivery channels, such as email, SMS, push notifications, and in-app notifications, to ensure that users receive notifications in a way that is convenient and effective for them.Reporting and analytics: The notification sender system will provide reporting and analytics capabilities to help the sender track the effectiveness of their notifications and make data-driven decisions about future notifications.

This work is organized by the following order: Chapter 1 serves as the introduction, offering an overview of the problem, who benefits from our program, and plan to solve the problem. In Chapter 2, we present a comprehensive review of existing solutions and relevant literature. Chapter 3 outlines our expected outcomes and objectives. Chapter 4 focuses on the work process, detailing the methods, tools, and analyses conducted. Lastly, Chapter 5 describes our proposed testing approach to validate the effectiveness of our program.

# Background

Notifications have become an integral part of today's digital landscape, as they significantly contribute to user engagement and retention in mobile applications [1]. With the ever-increasing number of apps vying for users' attention, leveraging effective notification management systems has become paramount. Personalized notifications tailored to user preferences and behavior have proven to be particularly effective in capturing users' attention and driving engagement. Research has shown that such targeted notifications lead to higher click-through rates and enhance overall user satisfaction [2].

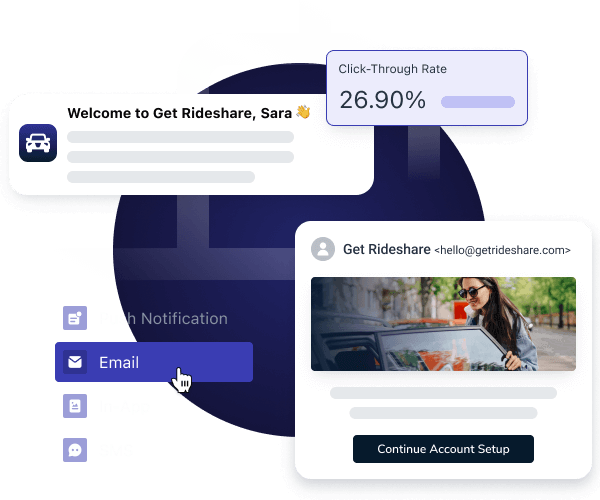
However, the timing and frequency of notifications play a crucial role in their effectiveness and user perception. Improper timing or excessive notifications can lead to notification fatigue and decreased engagement. Users may become overwhelmed and develop a negative association with the app, resulting in reduced interaction and even uninstallation. Studies have examined the impact of timing and frequency on users' cognitive load and perception, revealing that well-timed and carefully spaced notifications yield better outcomes [3]. Developers must strike a balance, considering the importance and urgency of the notification to avoid inundating users with irrelevant or untimely messages [4].

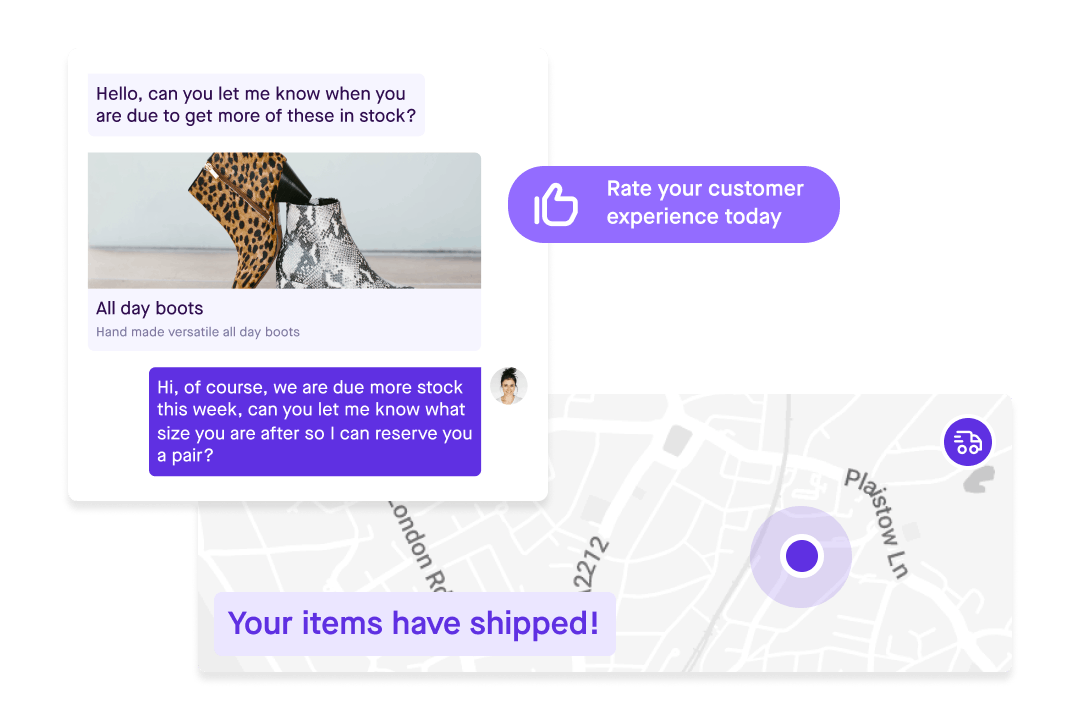
On the positive side, notifications serve as effective reminders for important tasks, such as appointments, deadlines, and events. By leveraging mobile notifications, users can stay organized and enhance their productivity. For instance, an intelligent system for reminder generation using mobile notifications has been proposed, allowing users to receive timely alerts for various tasks.5 This feature proves invaluable for individuals managing busy schedules and juggling multiple responsibilities.

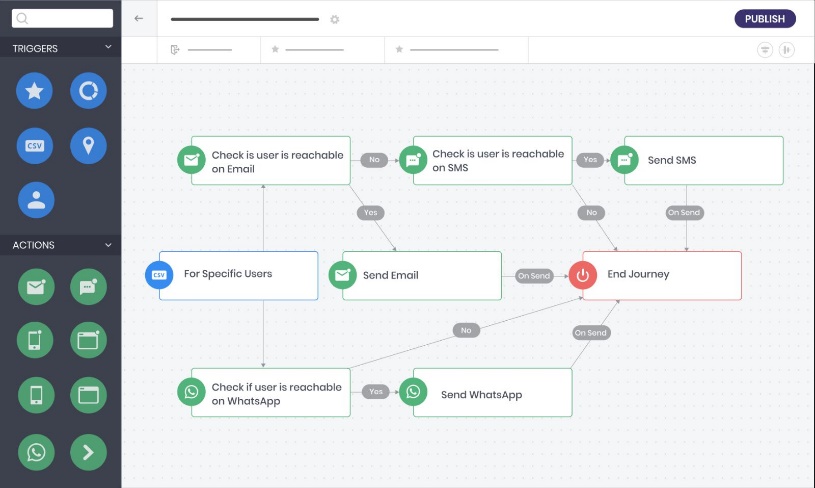
In conclusion, notifications have become a powerful tool in enhancing user engagement and retention in mobile applications. By leveraging personalized and well-timed notifications, developers can increase click-through rates, satisfaction, and overall user experience. However, it is crucial to find the right balance to avoid overwhelming users with excessive or irrelevant notifications. By carefully considering timing, frequency, and the importance of the notification, developers can harness the potential of notifications to provide valuable reminders and improve users' organizational skills and productivity.

## Existing solutions

Here we present two notification management systems as examples. These programs demonstrate different approaches and features that can enhance user engagement and improve the overall user experience.

OneSignal: A popular web push notification service “<https://onesignal.com/>” [6] that offers a user-friendly interface, advanced targeting and automation features, and seamless integration with a wide range of platforms and frameworks.

Pusher: Pusher is a cloud-based messaging platform that provides real-time web push notifications, as well as in-app notifications, chat functionality, and other messaging features. It offers a simple and easy-to-use API, as well as integrations with popular platforms such as Node.js and Laravel. Pusher provides real-time delivery and scalable infrastructure, making it a good choice for applications with high volumes of real-time data “ <https://pusher.com/>” [7].

WebEngage: WebEngage is a marketing automation platform that offers web push notifications, as well as in-app messages, surveys, and other engagement tools, to help businesses improve user experience and drive conversions. It provides advanced features such as journey mapping, user segmentation, and personalization, as well as integrations with popular platforms such as Shopify and Magento. WebEngage offers a range of pricing plans, from a free plan for small businesses to enterprise-level plans with custom features and dedicated support “ <https://webengage.com/>”[8].

## 2.2. The Evolution of Notification Sender Systems

Notification sender systems have come a long way in tandem with technological advancements and the evolution of communication channels. From the early days of telegraph and postal services, where messages were physically delivered, to the digital era, these systems have transformed significantly.

The advent of computing introduced email as a prominent communication method. Email notification management systems were developed to alert users about new messages in their inbox, utilizing protocols like SMTP for message exchange. As the internet became more accessible, web-based notification management systems emerged, allowing websites to deliver real-time updates directly to users' browsers through push notifications.

The rise of smartphones and mobile apps revolutionized notification management systems once again. Native push notification services on mobile operating systems enabled apps to send timely and personalized notifications to users' devices, even when the apps were inactive. Social media platforms also played a significant role by introducing sophisticated notification management systems that kept users informed about activities within their networks.

Furthermore, notification sender systems expanded beyond traditional channels. Messaging apps introduced real-time notification features for seamless communication and collaboration. Voice assistants and smart speakers incorporated notification capabilities, enabling users to receive spoken alerts and reminders.

The evolution of notification sender systems has been driven by the need for instant communication, personalization, and improved user experiences. Technological advancements in areas like cloud computing, artificial intelligence, and data analytics have further enhanced the capabilities of these systems.

Today, notification sender systems are crucial across various industries, empowering businesses to engage with their audiences, deliver important updates, and facilitate user actions. They have become integral to sectors such as e-commerce, news, social networking, healthcare, and finance.

Looking ahead, notification sender systems are expected to become even more sophisticated, leveraging emerging technologies like machine learning, natural language processing, and augmented reality. These advancements will enable them to adapt to evolving user preferences, communication trends, and the demands of an increasingly connected world.

# Expected Achievements

We have high expectations for our notification management system. We believe that it has the potential to significantly enhance our users' experience, improve engagement, and ultimately retain our users. We expect our notification management system to provide timely and relevant notifications that are triggered by specific events or actions, customized based on user preferences, and delivered through multiple channels, such as email, SMS, and push notifications.

We also anticipate our notification management system to be designed in a way that enhances the overall user experience. This includes making the notification settings intuitive and easy to customize, allowing users to snooze notifications or dismiss them easily, and ensuring that notifications don't disrupt the user's workflow. By designing our notification management system in this way, we believe that our users will be more likely to engage with our service and stay loyal to us.

Another requirement we have for our notification management system is that it facilitates better communication between our users and our service. We want our notification management system to provide users with updates about their account status, new features or content, and important events or deadlines, in a timely and personalized manner. This will help our users stay informed and engaged with our service, and ultimately lead to a better overall user experience.

Furthermore, we anticipate our notification management system to retain users by providing personalized and relevant notifications that keep them engaged with our service. By delivering notifications that are tailored to the user's preferences and needs, we believe that our users will be more likely to stay engaged with our service and find value in it.

Finally, we anticipate our notification management system to improve security by providing notifications for critical security events, such as failed login attempts or unusual account activity. This will help our users take proactive measures to protect their accounts and sensitive information, ensuring that our service remains secure and trusted by our users.

Overall, we are committed to designing a notification management system that meets these requirements and delivers value to our users. We believe that by doing so, we will be able to enhance our users' experience, improve engagement, and retain our users in the long term.

## 3.1 Criteria for Success

1. Reliability: The notification management system should be reliable in delivering notifications consistently and without delay. Users should have confidence that the system will deliver notifications promptly and accurately.
2. Scalability: The system should be able to handle a growing number of users and notifications without performance degradation. It should be capable of scaling up to accommodate increased usage and handle high volumes of notifications efficiently.
3. Customizability: Users should have the ability to customize their notification preferences according to their specific needs and preferences. The system should allow users to choose the types of notifications they receive, the frequency of notifications, and the preferred communication channels.
4. Multichannel Support: The notification management system should support multiple communication channels to reach users effectively. This includes email, SMS, push notifications, voice calls, and other relevant channels. The system should be flexible enough to adapt to different user preferences and technological advancements.
5. Personalization: The system should enable personalized notifications tailored to individual users. It should utilize user data and preferences to deliver relevant and targeted notifications. Personalization can enhance user engagement and satisfaction with the notification management system.
6. Security and Privacy: The notification management system should prioritize the security and privacy of user data. It should adhere to best practices in data protection, encryption, and secure transmission of notifications. Users should have confidence that their personal information is handled securely and that their privacy is respected.
7. Analytics and Insights: The system should provide analytics and insights on notification performance. It should track delivery rates, open rates, click-through rates, and other relevant metrics. These insights can help organizations evaluate the effectiveness of their notifications and make data-driven improvements.
8. Integration: The notification management system should integrate seamlessly with existing software systems and platforms. It should have APIs or connectors that allow it to interact with other applications, databases, or customer relationship management (CRM) systems. Integration capabilities enable streamlined workflows and data synchronization.
9. Monitoring and Alerts: The system should include monitoring capabilities to detect any issues or failures in notification delivery. It should provide alerts or notifications to administrators or technical teams in case of failures or abnormalities. Timely notifications of system issues help ensure prompt resolution and minimize downtime.
10. User Feedback and Support: The system should have mechanisms for users to provide feedback on the notifications they receive. It should also offer support channels for users to seek assistance or report any issues they encounter. Feedback and support mechanisms help improve the user experience and address any concerns or challenges.

# Engineering Process

## 4.1 Processes:

**4.1.1 Research and motivation**

To broaden our understanding of the notification management system, we embarked on comprehensive research to answer critical questions. Firstly, we focused on identifying the primary users of the system, ensuring that their roles, responsibilities, and needs are considered during the development process. This user-centric approach will result in a system that is intuitive, user-friendly, and tailored to their requirements. Additionally, we explored how the system will seamlessly integrate with existing tools and systems within the organization, such as email clients or project management software. By understanding these integration points, we can ensure efficient workflows and minimize disruptions. Furthermore, we thoroughly examined the organization's notification policies and procedures, including guidelines on frequency and types of notifications, as well as procedures for managing and responding to notifications. This knowledge will guide us in designing a system that aligns with the organization's established practices. Lastly, we placed significant emphasis on addressing security and privacy considerations. By incorporating robust security measures, protecting user data, and adhering to relevant regulations and standards, we aim to instill confidence and trust in the system.

**4.1.2 Work process**

Building a notification management system entails a series of essential processes to ensure its successful implementation. We begin by defining the requirements, meticulously identifying the notifications to be managed and the specific needs of the organization. With the requirements in place, we move on to designing the system architecture, making informed decisions on the technology stack and overall structure to optimize functionality and integration. Next, we engage in the development process, where our skilled team writes code, integrates systems, and conducts thorough testing to guarantee the system's reliability and performance. Once the system is developed, we proceed to its implementation within the organization. This involves configuration, user account setup, and comprehensive training to facilitate a smooth adoption process. To ensure the system's longevity, we emphasize ongoing monitoring and maintenance, proactively addressing any issues, making necessary updates, and addressing user concerns promptly. It is crucial to recognize that building a notification management system is an iterative process, with the need for continuous adjustments and updates to align with evolving organizational requirements.

**4.1.3 Constraints to be taken care of**

As we embark on the development journey of our notification management system, it is important to acknowledge and carefully consider the constraints that may arise throughout the process. These constraints will have an impact on our development approach, but with proper planning and proactive measures, we can navigate them effectively.

Time constraints will play a significant role in our development process. We need to establish realistic timelines and milestones, ensuring that we allocate sufficient time for each stage of development, including design, implementation, testing, and deployment. By closely managing our time and adhering to a well-defined development plan, we can meet project deadlines and deliver the system on schedule.

Technical constraints are another crucial aspect to address. We must assess the limitations and capabilities of our chosen technology stack, development tools, and infrastructure. This evaluation will help us make informed decisions and devise strategies to overcome any technical challenges that may arise. Regularly monitoring emerging technologies and industry best practices will enable us to adapt and leverage advancements to enhance the functionality and performance of our notification management system.

Compliance and regulatory constraints cannot be overlooked. Depending on the nature of the system and the data it handles, we must adhere to relevant legal requirements, data protection regulations, and industry standards. It is imperative to integrate privacy controls, security measures, and necessary compliance protocols into our development process. Regular audits and assessments will help us ensure that the system complies with all applicable regulations and safeguards user data appropriately.

Last but not least, user requirements and expectations should be our guiding light. We must actively engage with users, conduct user research, and gather feedback throughout the development process. By placing a strong emphasis on user-centric design and incorporating iterative feedback loops, we can develop a notification management system that caters to the needs, preferences, and evolving expectations of our users.

In summary, by recognizing and addressing these constraints throughout the development process, we can mitigate risks, deliver a robust and user-friendly notification management system, and achieve our development goals effectively. Flexibility, adaptability, and proactive decision-making will be key in navigating these constraints and creating a successful product that meets the needs of our users.

## 4.2 Product:

## 4.2.1 Requirements

**Functional:**

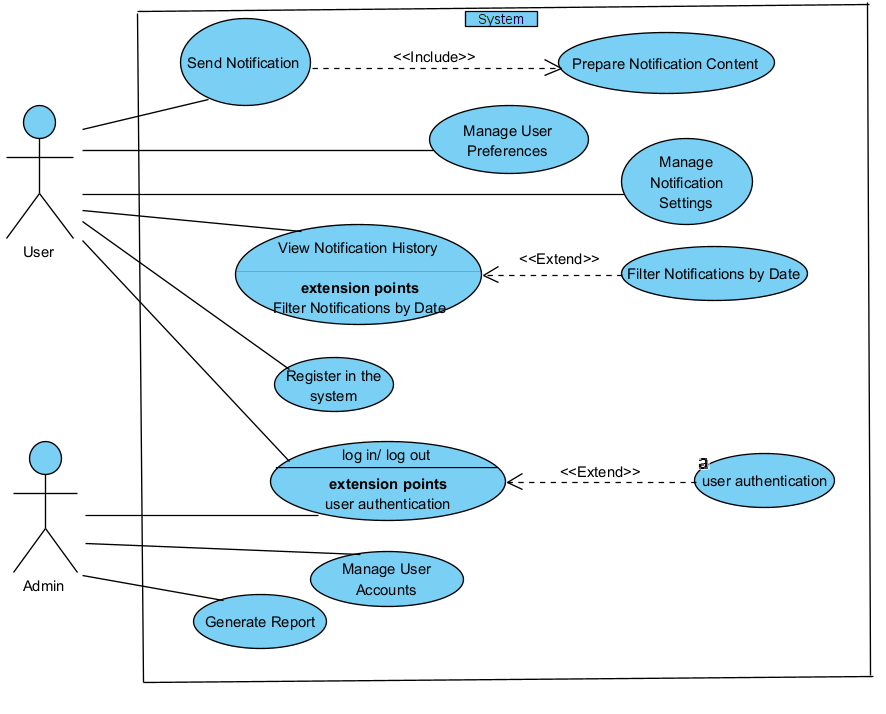
|  |  |
| --- | --- |
| **1** | The system should provide a user registration process to allow users to create accounts. |
| **2** | Users should be able to manage their profiles, including updating personal information and notification preferences. |
| **3** | The system should support user authentication and authorization to ensure secure access to the platform. |
| **4** | Users should have the ability to create and customize notifications, including defining the content, title, and associated actions |
| **5** | The system should provide options for different notification types, such as push notifications, emails, or SMS messages |
| **6** | Users should be able to schedule notifications for future delivery or send them immediately. |
| **7** | The system should allow users to select the target audience for their notifications based on various criteria, such as user segments, geographic location, or specific user attributes |
| **8** | Users should have the option to track delivery success rates and analyze recipient engagement metrics. |
| **9** | Users should be able to access metrics such as open rates, click-through rates, and conversion rates to evaluate the impact of their notifications. |
| **10** | The system should include an administrative dashboard to allow administrators to manage user accounts, monitor system performance, and handle any system-related issues. |

**Non-functional:**

|  |  |
| --- | --- |
| **1** | Highly reliable, ensuring that notifications are delivered without failures or disruptions. |
| **2** | Backup and recovery processes in place to safeguard against data loss or system failures. |
| **3** | The system should be able to scale up or down based on fluctuating usage patterns without compromising performance |
| **4** | User-friendly interface that is intuitive and easy to navigate |
| **5** | Clear documentation and user guides should be provided to assist users in understanding and utilizing the system effectively |
| **6** | Compatible with various platforms and devices to ensure a seamless user experience. |
| **7** | Narration support and texts for deaf people. |
| **8** | Simple interaction mechanism |

## 4.2.2 Diagrams

Use Cases:



* **Use case 1: (Send Notification)**

1. Goal: Send notifications based on specific triggers or events.
2. Precondition: User preferences for notifications are set.
3. Possible user errors: The notification is not successfully sent.
4. Pseudo code Flow:

|  |  |
| --- | --- |
| Actor | System |
| Click on “Send notification” | Open new page to prepare a notification. |
| Select the time and the date to send |  |
| Select the users who want the notification to reach. |  |
| Click “confirm/send” | Sent the notification by the instructions. |

* **Use case 2:(Manage user preferences)**

1. Goal: It includes actions like allowing users to update their contact information, language preferences, or any other relevant preferences that may impact their interaction with the notification management system.
2. Precondition: User are registers on the system.
3. Postconditions: The preferences are updated successfully.
4. Pseudo code Flow:

|  |  |
| --- | --- |
| Actor | System |
| Click on “Manage user preferences” | Open manage user preferences. |
| Update ... |  |
| Click “save” | Save the update page of manage user preferences. |
|  |  |

* **Use case 3:(Manage notification Settings)**

1. Goal: This use case allows users to customize their notification preferences according to their preferences. Users can select the types of notifications they want to receive (such as email alerts, system notifications, or reminders), choose their preferred delivery channels (email, SMS, push notifications, etc.), and set the frequency of notifications (immediate, daily digest, etc.).
2. Precondition: User is logged in and has access to notification settings.
3. Postconditions: User's notification preferences are updated and saved.
4. Pseudo code Flow:

|  |  |
| --- | --- |
| Actor | System |
| Click on “Manage notification Settings” | Open Manage notification Settings page. |
| Update ... |  |
| Click “save” | Save the update page of Manage notification Settings. |
|  |  |

* **Use case 4:(View Notification History)**

1. Goal: This use case enables users to access a log or history of all the notifications they have received. It allows users to review past notifications, including their timestamps, content, and status (e.g., read, unread, dismissed). Users can filter or search through the notification history and sort notifications based on criteria such as date or type.
2. Precondition: User is logged in and has received notifications.
3. Postconditions: User can view their notification history and interact with individual notifications if applicable (e.g., marking as read, dismissing).
4. Pseudo code Flow:

|  |  |
| --- | --- |
| Actor | System |
| Click on “View Notification History” | Open View Notification History page. |
| Click filter by date | Show the history by date. |
| Click “Return or close” |  |
|  |  |

* **USE CASE 5 : (Register in the system)**

1. Goal: The user create new user.
2. Precondition: User don’t have an account.
3. Possible user errors: The fields not full properly.
4. Limitation: Multiple accounts with the same email address.
5. Pseudo code Flow:

|  |  |
| --- | --- |
| Actor | System |
| 1)Click ‘sign up’ button. | 2)Display registration form. |
| 3) Fill the fields . |  |
| 4) Click ‘Register’ button. | 5) Validate the input. |
|  | 6) Display the login form. |

* **USE CASE 6 : (login)**

1. Goal: login in the system.
2. Precondition: There are no preconditions associated with this use case.
3. Possible user errors: User entered wrong password.
4. Limitation: Limit Login Attempts for login protection.
5. Pseudo code Flow:

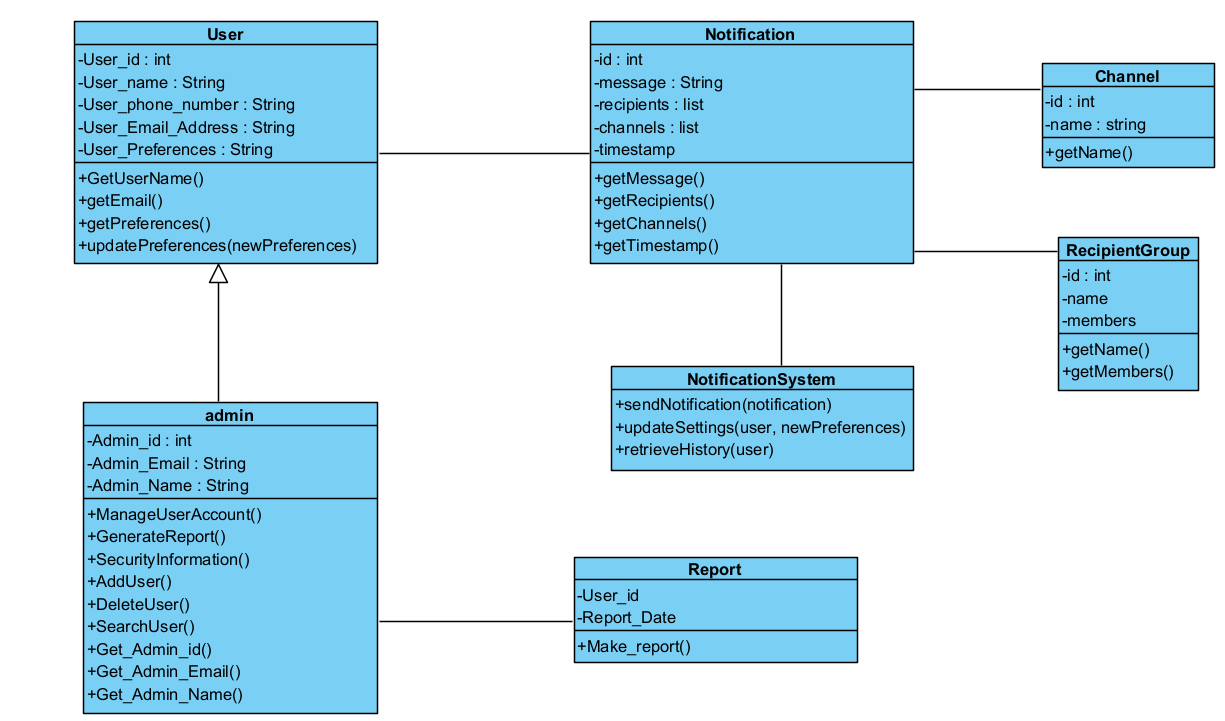
|  |  |
| --- | --- |
| Actor | System |
| 1) Click ‘login’ button. | 2)Display login form. |
| 3) Types his/her name and password. | 4) Validates password. |
|  | 5)Logs him/her into the system. |

* **USE CASE 7 : (Register in the system)**

1. Goal: The user create new user.
2. Precondition: User don’t have an account.
3. Possible user errors: The fields not full properly.
4. Limitation: Multiple accounts with the same email address.
5. Pseudo code Flow:

|  |  |
| --- | --- |
| Actor | System |
| 1)Click ‘sign up’ button. | 2)Display registration form. |
| 3) Fill the fields . |  |
| 4) Click ‘Register’ button. | 5) Validate the input. |
|  | 6) Display the login form. |

**Class Diagram:**



1. Notification Class:

* Description: Represents a single notification message.
* Attributes:
  + **id**: Unique identifier for the notification.
  + **message**: Content of the notification message.
  + **recipients**: List of users or recipient groups who will receive the notification.
  + **channels**: List of communication channels through which the notification will be sent.
  + **timestamp**: Date and time when the notification was sent.
* Methods:
  + **getMessage()**: Retrieves the content of the notification message.
  + **getRecipients()**: Retrieves the list of recipients for the notification.
  + **getChannels()**: Retrieves the list of communication channels for the notification.
  + **getTimestamp()**: Retrieves the timestamp of when the notification was sent.

1. User Class:

* Description: Represents a user of the notification management system.
* Attributes:
  + **id**: Unique identifier for the user.
  + **username**: Username of the user.
  + **email**: Email address of the user.
  + **preferences**: User preferences for notification settings.
* Methods:
  + **getUsername()**: Retrieves the username of the user.
  + **getEmail()**: Retrieves the email address of the user.
  + **getPreferences()**: Retrieves the user's notification preferences.
  + **updatePreferences(newPreferences)**: Updates the user's notification preferences.

1. Administrator Class:

* Description: Represents an administrator or system operator.
* Attributes:
  + **id**: Unique identifier for the administrator.
  + **username**: Username of the administrator.
  + **email**: Email address of the administrator.
* Methods:
  + **getUsername()**: Retrieves the username of the administrator.
  + **getEmail()**: Retrieves the email address of the administrator.

1. NotificationSystem Class:

* Description: Manages the sending and processing of notifications.
* Methods:
  + **sendNotification(notification)**: Sends the specified notification to the recipients through the configured channels.
  + **updateSettings(user, newPreferences)**: Updates the notification preferences of a user.
  + **retrieveHistory(user)**: Retrieves the notification history for a user.

1. Channel Class:

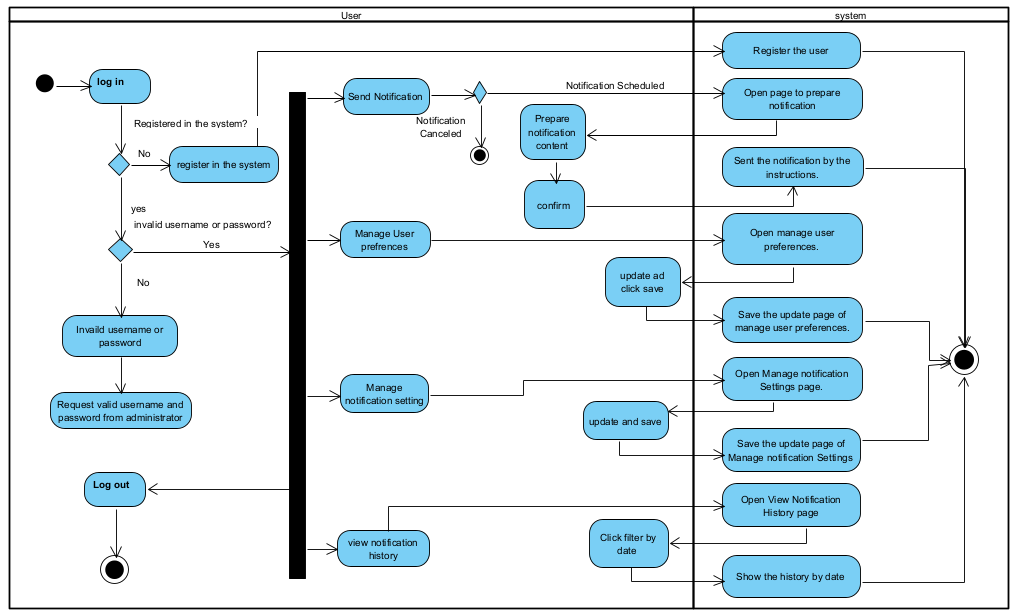
* Description: Represents a communication channel through which notifications can be sent.
* Attributes:
  + **id**: Unique identifier for the channel.
  + **name**: Name or identifier of the channel.
* Methods:
  + **getName()**: Retrieves the name or identifier of the channel.

1. RecipientGroup Class:

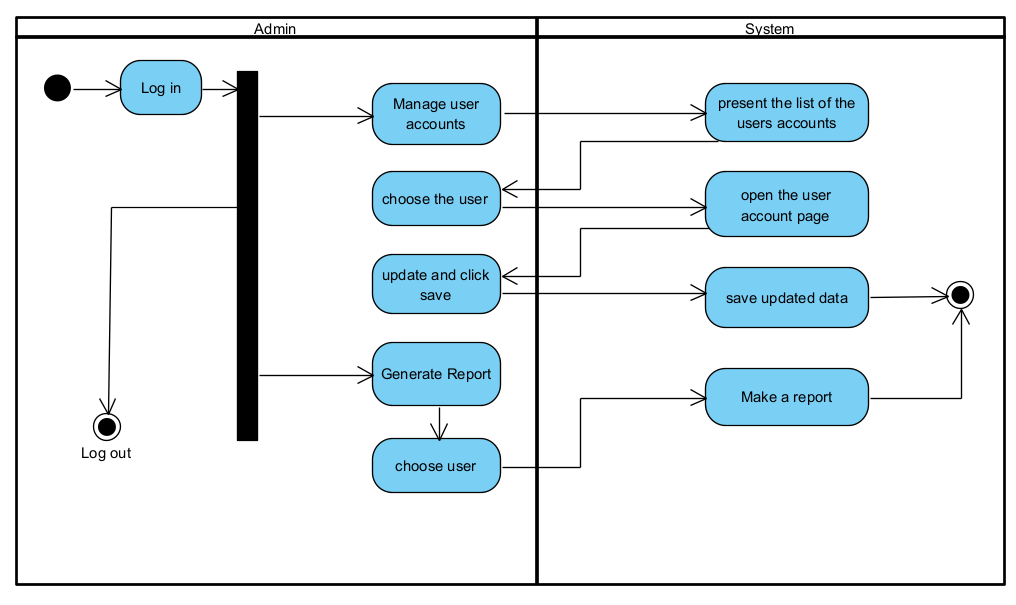
* Description: Represents a group of users who will receive a notification.
* Attributes:
  + **id**: Unique identifier for the recipient group.
  + **name**: Name or identifier of the recipient group.
  + **members**: List of users who are members of the group.
* Methods:
  + **getName()**: Retrieves the name or identifier of the recipient group.
  + **getMembers()**: Retrieves the list of users who are members of the group.

**Activity Diagram:**

**Activity diagram between User and system.**



**Activity diagram between Admin and system.**



# 5.Testing

*Login and user registration*

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Expected Result | Pass/Fail |
| 1 | Start system | Main system page opened | Pass |
| 2 | Enter the correct user name and password. then click login button | Accept user Name and password and display | Pass |
| 3 | Enter the invalid Username and Password and click login button | Error message “Login Failed” | Pass |
| 4 | In adding form enter new details click Submit button | Added user Contact detail to database. | Pass |
| 5 | Signup new user by using Register form Clicking Signup button | Appear login form | Pass |
| 6 | In change of adding Details to register Form When enter invalid Email address unequal password for Signup | Display “Enter valid Email Address and passwords should be same. | Pass |
| 7 | Resetting password by using login form Clicking Forgot Password button | Appear resetting password form | Pass |
| 8 | Enter the Email Address or Phone number and click Continue button | Accept Email Address or Phone number and display resetting option | Pass |
| 9 | Resetting passwordbyEntering new password and confirmation code Clicking Reset Password button | Appear login form | Pass |

*User main actions*

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Expected Result | Pass/Fail |
| 1 | Enter the correct user name and password. then click login button | Appear main page | Pass |
| 2 | Options Menu by Clicking Menu button | Appear Menu Options | Pass |
| 3 | Main Window Option by Clicking Main Window Item | Appear main page | Pass |
| 4 | Logout Option by Clicking Logout Item | Appear login page | Pass |
| 5 | About us Option by Clicking About us Item | Appear about us page | Pass |
| 6 | Update details Option by Clicking update details Item | Appear update user details page | Pass |
| 7 | Enter new user name. then click update button | Username updated | Pass |
| 8 | Enter old password and new password. then click update password | Password updated | Pass |

*Admin main Actions*

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Expected Result | Pass/Fail |
| 1 | Manage user accounts by using update user details page Clicking search button | found specific user | Pass |
| 2 | Manage user accounts by using update user details page Clicking delete button | Delete specific user | Pass |
| 3 | Manage user accounts by using update user details page Clicking update button | update specific user detail | Pass |
| 4 | Manage user accounts by using update user details page Clicking add button | Add new user | Pass |
| 5 | See all users details by clicking manage user account button | Appear manage user account page | Pass |
| 6 | Enter the correct user name and password. then click login button | Appear Admin main application page | Pass |
| 7 | Enter the invalid Username and Password and click login button | Error message “Login Failed” | Pass |
| 8 | Update Admin details by clicking Security info button | Appear update user details page | Pass |
| 9 | Enter Old Password and New Password and click update Password button | Password updated | Pass |
| 10 | Generate reports by clicking generate reports button | Appear generate reports page | Pass |
| 11 | Fill correct report date and click search button | Report found | Pass |
| 12 | Fill incorrect report date and click search button | Error message “Report not found” | Pass |
| 13 | Generate new report by clicking generate report button | new report generated | Pass |

*Send notification*

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Expected Result | Pass/Fail |
| 1 | Click “send notification” | Open send notification page | Pass |
| 2 | Choose time in the past to send the notification in it | Error “the time is not relevant | Pass |
| 3 | Choose time of the day but with past hours | Error “the time is not relevant | Pass |
| 4 | Do not choose the people who will receive the message | Error message “choose people to receive message” | Pass |
| 5 | Click confirm | Message that the notification sent successfully | Pass |
| 6 | Sent notification without content | Error message ”please fill the content of your notification” | Pass |

# 6.References

[1] Walia, N., & Sharma, S. (2018). Enhancing User Engagement and Retention in Mobile Applications through Push Notifications. International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), 3(4), 1625-1632.

[2] Li, J., & Mao, J. (2018). User Behavior-Based Personalized Mobile Push Notifications. Mobile Information Systems, 2018, 1-11.

[3] Huang, H. C., Li, Y. C., & Chiu, M. L. (2020). Understanding the Effect of Timing and Frequency of Smartphone Notifications on Users' Cognitive Load and Perception. Telematics and Informatics, 55, 101418.

[4] Huang, Y. H., Huang, C. C., & Chen, Y. F. (2018). Examining the Impact of Smartphone Notifications on Consumers' Mobile Application Usage: Moderating Roles of Notification Importance and Time Sensitivity. Journal of Interactive Marketing, 44, 144-156.

[5] Ranjan, S., Ranjan, R., & Chatterjee, A. (2018). An Intelligent System for Reminder Generation Using Mobile Notifications. Journal of Ambient Intelligence and Humanized Computing, 9(2), 351-367.

[6] OneSignal. [Online]. Available: https://onesignal.com/. [Accessed: May 30, 2023].

[7] Pusher. [Online]. Available: https://pusher.com/. [Accessed: May 30, 2023].

[8] WebEngage. [Online]. Available: https://webengage.com/ . [Accessed: May 31, 2023].

ראש הטופס