Rowan Notification Application

Project Design Document

Written By Collin Wisser

Team: Kyle Takach, Chris Hicks, Ed Callihan Shayne Colomy, Collin Wisser

03/10/2020

1. **Introduction**

Rowan University notification application is a tool for Rowan University students to receive critical information from the university in a timely manner. This system is designed to shorten the complexity of communication between the university and the student by directly notifying the student when ever a critical email is received. This system also gives the user a level of freedom and customization for receiving these notification by giving the user the ability to add and drop email that they deem important, as well as adding keywords to an existing email in their list to narrow down the notifications even more. The key features of this application are as followed:

* Critical email notification
* Add/Drop email addresses
* Conditional/Absolute email notifications(keywords)
* Push to screen notifications
* Notification storage

This email notification app will be developed for cross-platform use on, Android and IOS mobile devices. With this said the majority of the user interface will be written and developed in Dart using Flutter (Application development SDK). The use of flutter will give this application the architecture needed to perform in a cross-platform environment as well as making upkeep and further development easy.

This deliverable is intended as a high-level software design document that can be used to help implementation of this application. The goal is to present a comprehensive blueprint on how this application will be designed and developed. This document will provide concise diagrams and explanations that highlighting the applications key features as well as a in depth view on the design and implementation.

1. **Application Overview**

The objective of this notification app is to lessen the complexity of the current communication system at rowan university by providing a notification system that alerts users of critical information sent via email or emails that the user as deemed important. As well as have a level of customization and a number of useful features to provide a simplistic and easy to use framework for the user. University administrators will also have configurable options for changing key functionality to the applications.

* 1. Design Overview
* Application platform: This application will be designed for use on both android and IOS operating systems for mobile devises. Having this systems user interface written in Dart and majority of the backend written in java gives this application the flexibility for cross platform accessibility.
* Application features: Features include
  + Push Notifications
  + Add and drop email addresses from a list
  + Add and drop keywords to a corresponding email address
  + A file system that stores notifications according to email address
  + A file system that stores notifications according to email address and corresponding keywords.
  + A push button that redirects the user to the location of the email received.
  + A snooze button that snoozes a notification for a user defined amount of time.
  + A button that adds a user’s instructors email addresses from that semester
  + The list of instructor email addresses are deleted after an arbitrary amount of time after the semester end date.
* Application user interface: The design and functionality of the user interface will be developed in Android Studio using googles Flutter SDK for cross platform development. This user interface will be a simplistic easy to use design that will encapsulate all the functionality discussed in the previous bullet.
  1. System Overview
* Application development building blocks: This applications backend will be written in java, using a JavaMail API for connecting and receiving information from an email service. The front end of this application will be written in Dart and be using Flutter SDK.
* Application dependencies: This application is being developed using JavaMail API and Flutter SDK for mobile devices, with this said to run and or test this software’s user interface and subsequent features you must have access to a mobile device running Android or IOS operating systems. Alternatively, desktops emulators can be used.
* Application Security: The login process for this application will be done through rowan universities CAS system. This system will forward the user to a rowan login page and verify the user’s credentials.

1. **System Architecture**
   1. Architectural Design: The principal around our design is to achieve an efficient system for notifying users on email that are important to them. Therefore, we focused our system around what the user wants to see and built from there.

The block diagram below shows the principal parts of the system and their interactions.

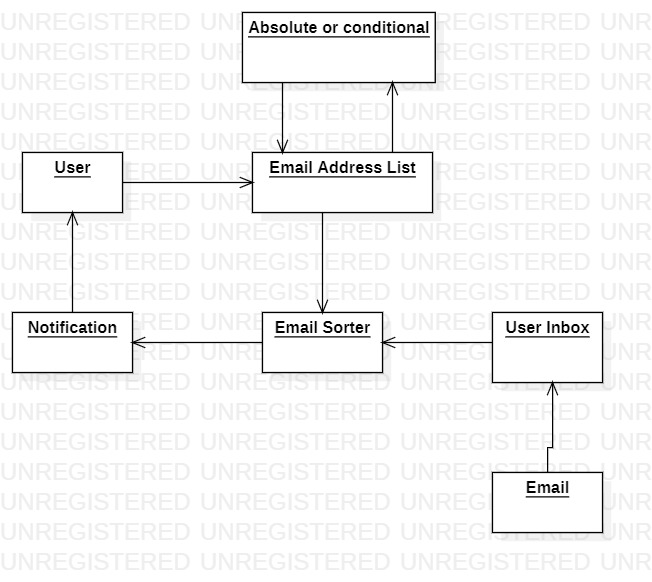


Figure 3.1 Block Diagram

* 1. System Design: Our system design revolves heavily around sorting emails from the inbox of the user, as well as keeping track of the emails we have already searched. This system records the time a search on a user’s inbox was done and updates every search, this is done to remove the possibility of duplicate notification.

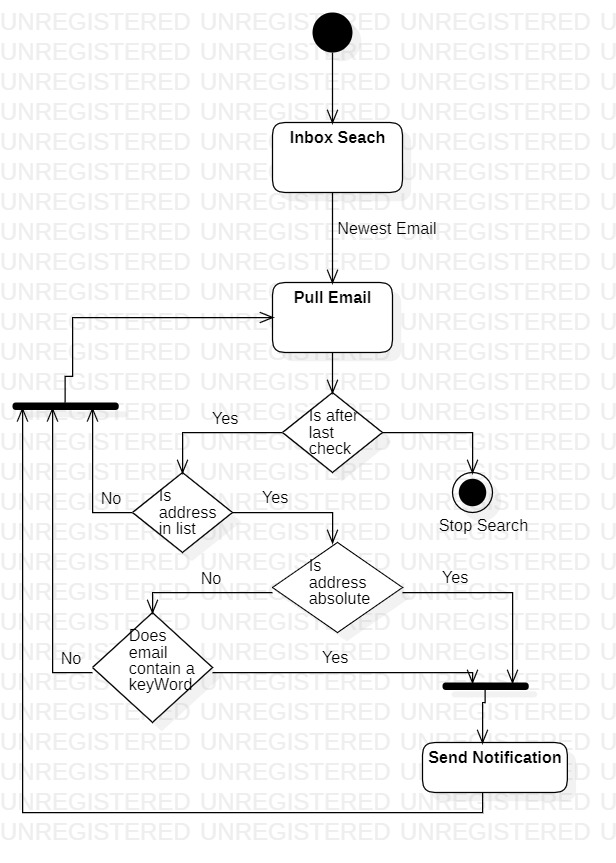
The following activity diagram will show a high-level idea of the sequence and logic of the searching system.

Figure 3.2 Activity Diagram

This system relies on periodic searches of a user’s email. The time in between the searches is arbitrary and will be a configurable variable.

The following diagram shows the periodic searching and its logic.

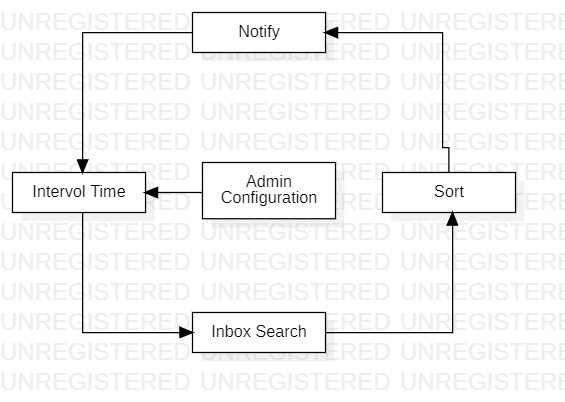


Figure 3.3 Search Configuration Diagram

Physical Diagram of CAS login and amazon webserver connection.

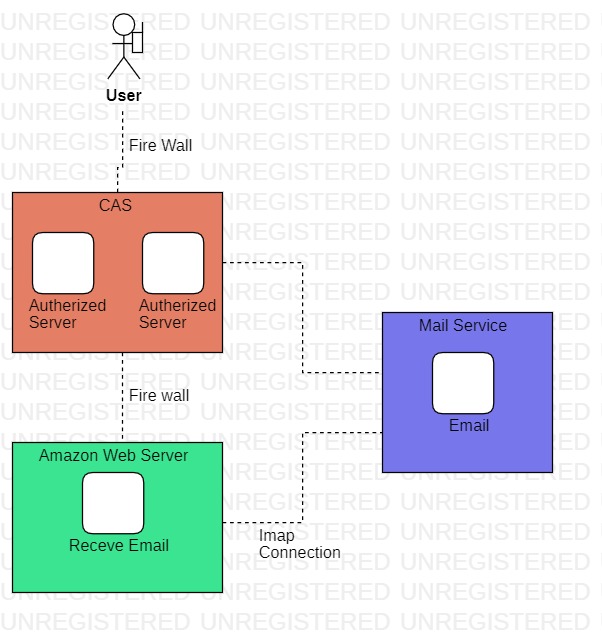
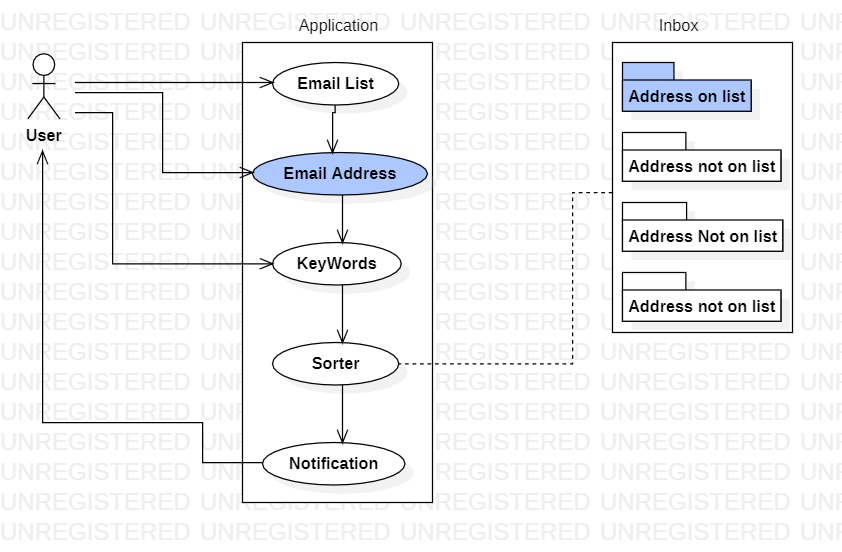


Figure 3.4 CAS connection diagram

1. **Use case**

A student Joe receives 10-15 emails a week from his university. Joe only wants to be notified about 1-3 of them due to them being from important people like professors and advisors, or they contain time sensitive information. Joe wants a system that will notify him whenever he receives an email from a email address that he deems important, as well as giving him the option to only receive a notification if that email contains a topic that he wants to see along with the address.

The diagram bellow describes the user’s interaction with the system and its interactions with the inbox and application.

Figure 4.1 Use Case Diagram

1. **Data Design**
   1. Data Description: Much of the data design is focused around storing user input such as the email list as well as storing notifications for later viewing. Other important data being stored is system settings, an internal time check, semester start and end date, and keywords.

The following data and their description are a high-level view of their role in the functionality of this application:

* Email List
  + A user will enter a number of email addresses to the email address list.
* Key Words
  + Key words are a functionality to help narrow down and filter emails from a specific address to find a desired topic or focus of an email.
    - There is currently a 5 keywords limit to any give email address
    - Once an email address is entered into the email list the user will have the option to enter keywords to make the notification conditional.
* Notifications
  + Notification will be stored in a file system with the email sender as the parent directory. Once a notification is received it will be stored in the file that was created for that email address. These notifications contain
    - Sender address
    - Email subject
    - Snippet of email body
    - Time and date sent
* File System
  + Each email in the user defined email list will be given a file directory for storing subsequent notifications. With in a file for an email address is will also contain keywords that the user has defined.
* Time Check
  + Once the search time is elapsed the system will search the user’s inbox and record the time it was done.
* User Settings
  + User settings will be stored once changed from default. As well as giving the user the option to revert to default settings.
* Semester Start/End date
  + This system will have a configurable option for and administrator to set the semester start and end date. This data will be used for adding and dropping user instructors.
* User Instructor list
  + This list will be separate from the email list due to its dependency on the semester start and end date.
    - Once the time check elapses the end date of the semester the email list will be removed from the application.

1. **Component and Module Description**

6.1 File System**:** This file system was designed to be simplistic and easy to use, trying to cut down as much time searching for a notification as possible. Once a email is added to the email list a parent directory is created with the email address as the name. this directory stores all notification received from that email address and is stored for a user defined amount of time. Once that time is elapsed, the notification will be deleted.

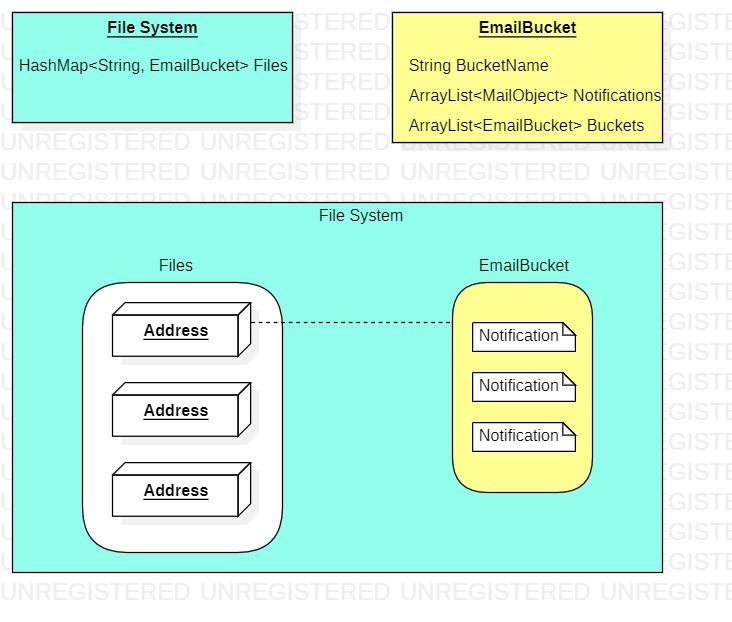
The object diagram bellow shows the logic of the file system and its data structures.

Figure 5.1 File system diagram

6.2 Notification: A notification is comprised of two parts, the push to screen notification and the stored notification within the application.

* Push to screen notification: This notification is displayed on the user’s mobile device screen, consisting of email sender, subject, body snippet and the date it was send. It will also have two functionalities, a button to snooze the notification to be redisplayed in a user defined amount of time, as well as the ability to push the notification and be sent to the native location of the email.
* Stored notification: The second notification type is stored internally, consisting of the same information as the push to screen notification. This notification only has one functionality and that is a push button that redirects the user to the native location of the send email.

6.3 Instructor Email List: The instructor email list is a separate list of email addresses that pertain the individual users’ instructors. With this said the application will need access to the individual user’s information on their instructors for that specific semester, this information will be accessed through rowans internal network banner. Once the application requires the necessary information it will give the user the option to app all their instructors emails to the email list to be notified. This email list will be implemented using a HashMap<String, String> with the key being or’ed strings.

6.4 Email list and Keywords: A core feature of this application is the ability to add and drop emails from a user defined email list. Any email address added to this will be by default absolute and the user will receive a notification once an email is received from that address. The user will also have the option to add keywords to a specific email address making this address conditional. A user may add up to 5 keywords and is designed to help narrow down the scope of the email search and give the user a level of configuration. Once a user has added a keyword to an email address, they will only receive a notification if an email is received by that address and the email contains one of the keywords defined, either in the body of the email or the subject.

The diagram bellow shows the logic and data structure of the email list and its keywords. The data structure used implementing the email list is a HashMap<String,String>, the key for the HashMap is strings separated by or’s and accessed by an xor operation.

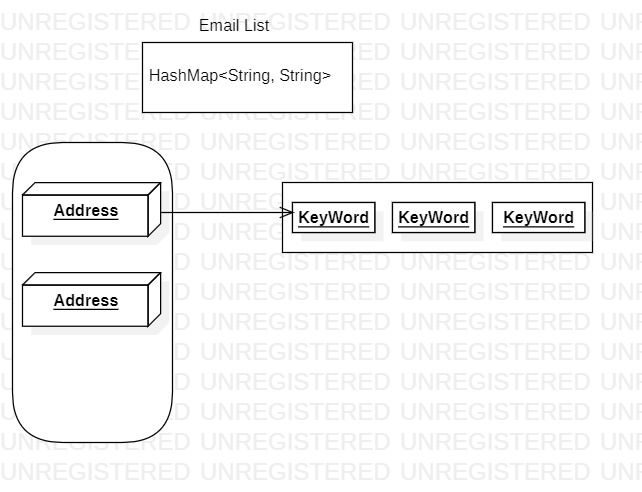


Figure 5.2 Keyword Diagram

6.5 Native email interaction and connection: As stated earlier in this document this system will be relying heavily on JavaMail API for the email connection and filtration. JavaMail will be providing this application with a direct connection to the user’s email. Our system will be using an IMAP protocol to receive the emails from the native inbox.

This diagram bellow shows the architecture of the JavaMail system and its interaction between the application.

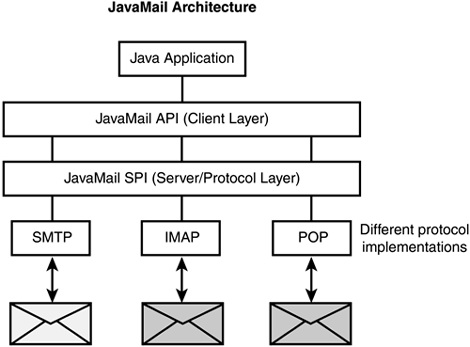


Figure 5.3 JavaMail Architecture Diagram

6.6 Email Filtration: The email filtration is essential for notifying the users only on their defined email addresses and the conditions they set for them. This applications filtration system is separated into multiple parts stating with:

* Searching The inbox: Once the proper credentials are verified and the application is given access of the user’s inbox a time check is recorded, and the application will only search through new emails. If a search finds an email after the time check it will proceed to pull information needed to verify that the email meets the right requirements.
* Check if valid sender: Once a new email is received the sender of the email is pulled and compared to the emails on the email list. If the sender is on the email list the system will check the email on the list is absolute or conditional. If the address is absolute the user is notified.
* Key-Word search: If the corresponding email address is conditional the system will proceed to search the subject and body of the email for any of the keywords associated with that address. If one is found, the user is notified

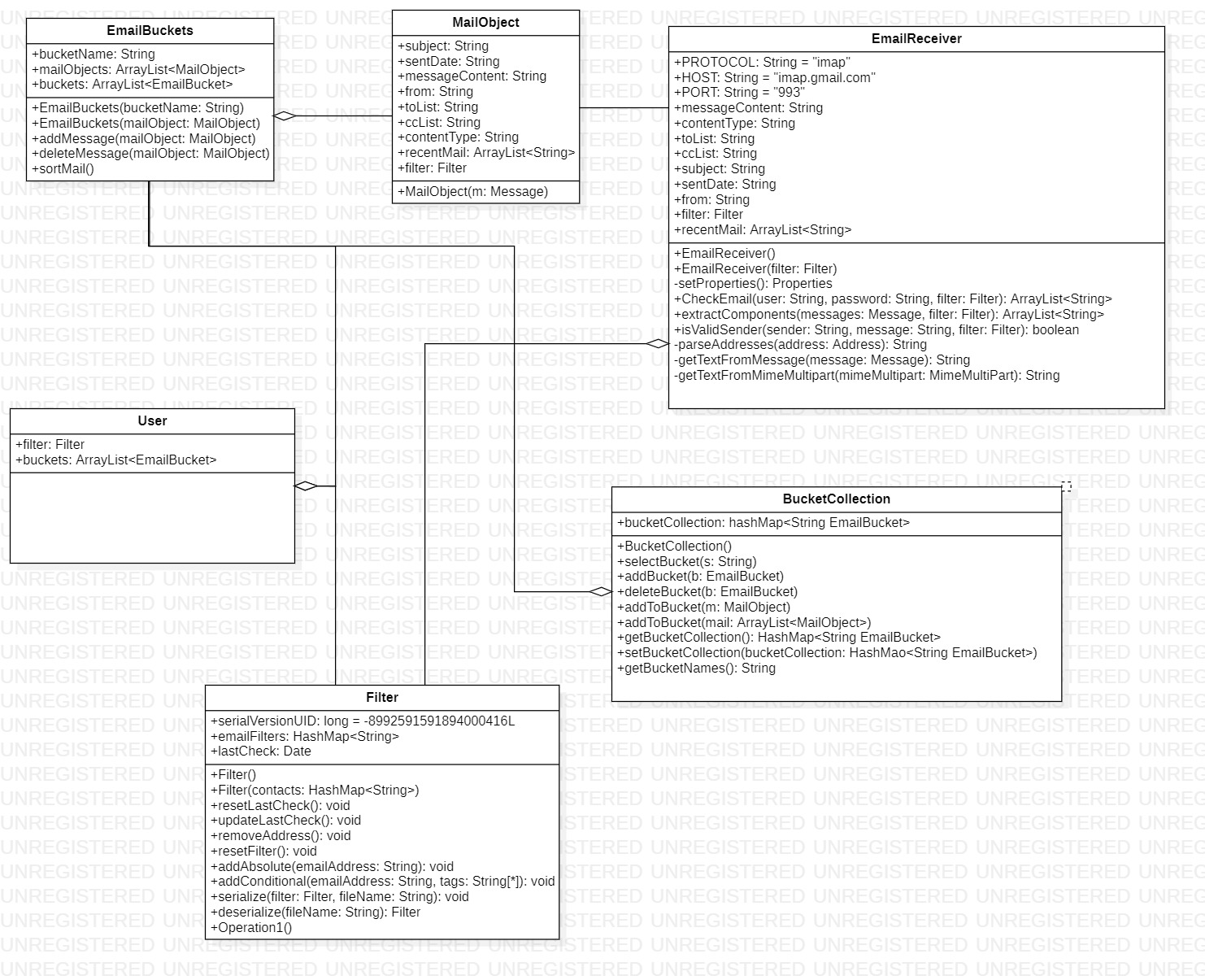
Class diagram:

Figure 6.3 Class Diagram

Class **EmailReceiver**: This class handles all the native email interaction and obtains the necessary information for the notification and sorting.

Class **Filter**: This class handles stores the information for filtering the emails from the inbox. This class contains the email list and keywords associated with the address.

Class **MailObject**: This class contains the information needed for the notification, such as the subject, sender and body snippet.

Class **EmailBucket**: This class is where the notification is stored and its correlation with an email address.

Class **BucketCollection**: This class is the location of the file system, each Emailbucket is stored within the BucketCollection with is corresponding email address.

Class **User**: The user class is used as a user object containing the filter object as well as the buckets

1. **Human Interface Design**
   1. UI design/Functionality: Our goal primary goal for our user interface is to make the experience as quick as possible while also encapsulating of the features we set out to implement. This system is design to cut out complications with communication through email and our user interface is not exception. We designed a 3-page user interface that houses all the necessary features as well as a push notification. The pages include Home Page, Settings and File’s page.

* Home Page: Our Home page is where the user would be spending most of their time on this application. This page houses several features, these are
  + Add/drop emails from email list: The user will have a button that will give them the option to add and drop emails from the email address. This button will prompt and ask for the email address and a name they would like to associate that address with.
  + Add/drop keywords: Once an email is added to the email list you can click on the list of email and a corresponding email address and an option will be available to add up to five keywords. Once there is at least one keyword it will be displayed with the option to remove it.
  + Settings button: This button will redirect the user to the settings page.
  + Pinned Notification’s: This will be a window that shows a notification that a user has decided to pin
  + Snoozed Notifications: The snoozed notifications will be displayed in a scroll window on the homepage, with the ability to click on it and be directed to the native email location.
  + List of all Email address files: This will be the location of the email address directory will be. We will display each directory with their corresponding names and have the ability to push on one and be directed to the files page that houses the notifications from that email.
* Settings Page: The settings page will contain all the necessary setting options for this application. These are
  + Updating instructor emails: This will give the users the option to add all their current instructors’ emails and edit them if needed.
  + Changing snoozed time: This will affect the amount of time a notification is snoozed before it alerts the user again.
  + Changing notification lifetime: This setting will affect the time a notification is stored in its directory.
* Files Page: This page will consist of all the notification for a specific email on the email list. Each email on the email list is given a directory (File Page) that will contain all the notifications created for that address. This page will list all the notifications created in a most recent order. These notifications will consist of the subject, body snippet, sender name, and date and time of the email sent. Clicking on a notification will result in being redirected to the native email location on the users email service. Each notification will have the option to be deleted from this list along with the option to pin it to the home page.
* Notification: Once an email is sent and meets the requirements set by the user, a push notification will be sent to the user’s screen in the notification section of their phone. This notification will contain the following information, the subject, body snippet, sender name, and date and time of when it sent. These notifications will also contain 2 functionalities. Once pressed on it will direct the user to the native email location. The second functionality is a button to snooze the notification for a user defined amount of time.

The Figure bellow shows a rough design of the home page and some of its features. Much of the User interface’s design and look is still under discussion but all the functionality is decided and ready for implementation.

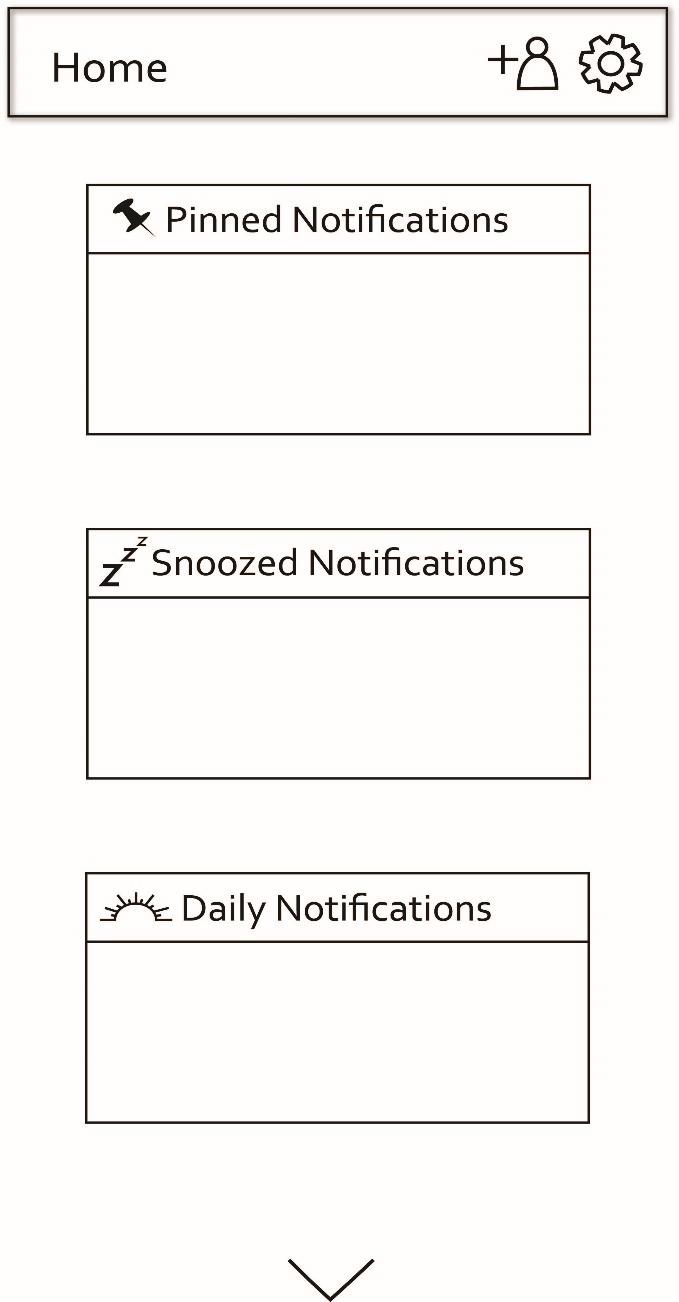


Figure 7.1 Home page Diagram

**Conclusion**

As a team we feel that this project has great potential to improve the connectivity of students and the university as well as improve the communication medium. We are striving to push the possibilities of this application and perfect the already existing ideas in place. With this project simplistic design gives us the ability to fine tune the features and helps cut the time of development down, as well has make it easy to continue working on this project. In the end our client will be maintaining this system so having a simplistic design was a key focus for our design decisions.