4 Particular aspect of our implementation : Email Engine

4.1 Requirements

The main goal of this engine is to respond to the clients' needs for having direct access to the ID cards of the users, which are sent by the users after registration. Indeed, the users are asked to send their ID cards through email to become validated and known users. Based on the clients' demand, we considered the following scenario to provide our clients with a fast and cheap solution that will be independent of any change in the platform over time.

4.1.1 Scenario

After a user registers on the platform, s/he sends an email containing their ID card. Since many users may register in a short time, the system should distinguish the received emails and recognize which email belongs to whom. To materialize such a distinguishability, the system should utilize the email that the users entered during registration to make a comparison between the received email and the emails recorded during registration. (It is worth mentioning that, to make this procedure feasible, we should have access to the mail server or at least to the inbox of the mail meant to receive the users' ID cards. However, Accessing Mail Inbox through IMAP protocol is feasible) According to this comparison, if there is a match with the received email, then the server should send the email to the admin platform and notify the Administrator of an email receipt for user Y. Furthermore if the user is already validated, the server should ignore it, and the process will consequently terminate. Validating and checking the ID card is accomplished by the Administrator, and if the ID card is approved, the status of the user should change to validated in the Database accordingly.

4.2 Design diagrams

Based on our scenario, we provide two diagrams (Sequence and Activity Diagrams), which depict an abstraction of the whole procedure and our engine.

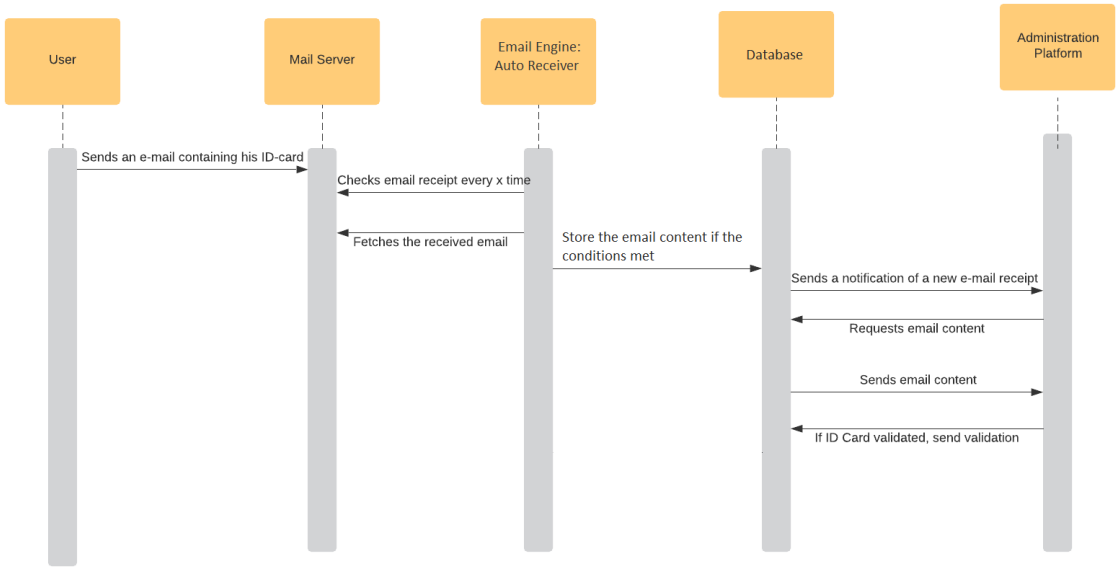

Figure 5: Sequence Diagram

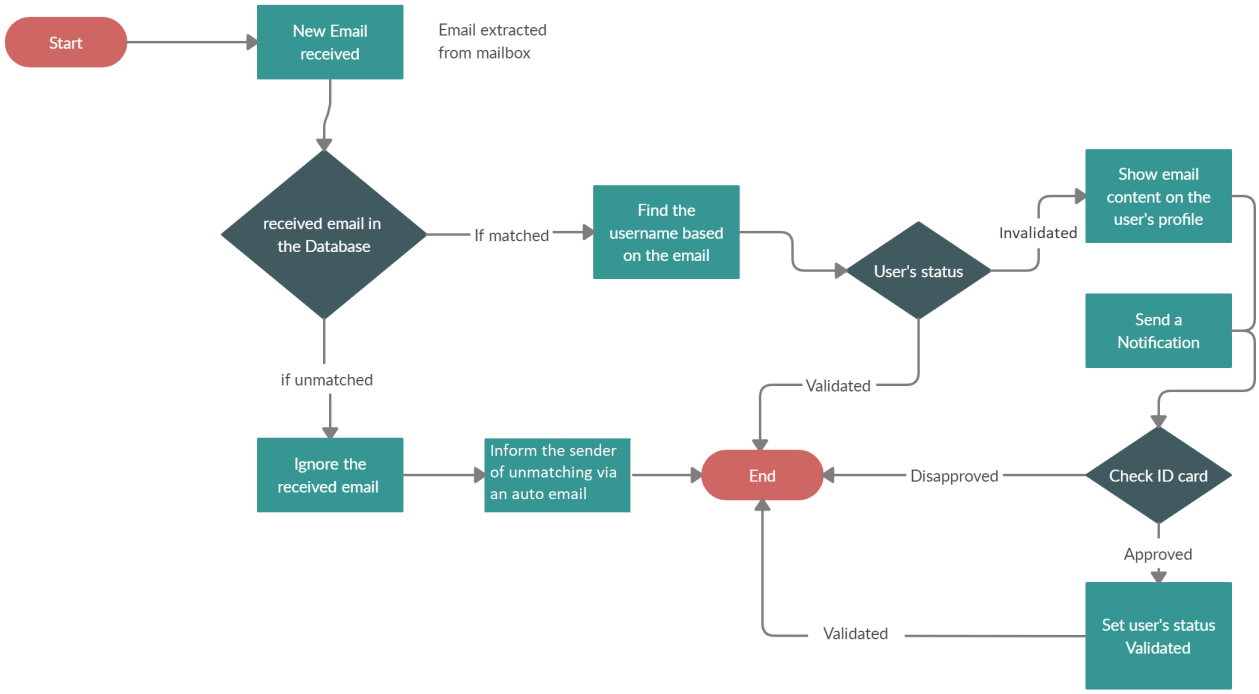

Figure 6: Activity Diagram for Email Engine

4.3 Implementation choices

Based on our scenario, we have explained that to make access to the email contents feasible, which are images in our case, we first need to read and fetch the unseen emails iteratively. Then if the sent email meets the criteria (e.g., the sender has yet not been validated/approved, the sender has already registered in the medium, etc.), then we should submit the email contents to the administration platform to enable the admins to check the identity of the users based on provided images in the email. In summary, this task is technically divided into two subtasks— one for receiving and processing emails in the backend and one for showing the email contents and notifying administrators in the frontend. The first task can be independently accomplished anywhere.

To this end, we designed an independent engine, trigger, email filtering using python 3.6.4. This engine can be run anywhere, on a personal computer, on Azure service as trigger, and so on; therefore no extra cost will be incurred and a high efficiency can be achieved.- Interested reader may refer to this link (click here) to know more about triggers. This engine can receive several files in the formats of .png, .jpg, and pdf, even in a single email. Then it extracts all images (for example, images from .pdf file) and embeds them in a single HTML file using a binary encoder. In this case, no pre-trained modules or python packages are used, which means that the process is quickly performed only through email protocols.

After extracting images, the generated HTML file is stored in the database so that the admin can simply load it. Technically speaking, to make it feasible, two new attributes, namely “status”, and “userIDimages”, must be added to the user table. “status” can hold three integer values, each of which shows the current approval status of the user. For instance, 0 is for WaitingForValidation/Approval, 1 is for Approval, and 2 is for Disapproval. Moreover, UserIDimages is a blob-type attribute that holds the HTML code that contains all id card images (images are embedded in this html file with binary

code).

A reasonable approach to avoid weighting the user table might be the use of an auxiliary table that only stores and HTML files. In this case, in the user table, the primary key of each file should be stored as a foreign key. In the frontend, after administrators receive a notification of an email receipt for user x, the HTML file related to the user should be upladed. This HTML file includes all details regarding the user. Here, we have attached an example of the HTML file that a user received via an automatic replying email as a confirmation after sending his/her images.
- Click here to see the corresponding commit and the example -

4.4 Testing Strategy for the link to the ID card

To test the link towards the email received that contains the ID card of the person, it is not possible to do unit tests because it is hard to have a function that can't check if a page opens. So we do the test manually by putting a URL link in the database and clicking on the button, then checking that the correct page opens.

To check the engine, we wrote a function that iteratively sends different files with different formats to the email that is meant for receiving the users' email. The number of files was about 50. At the same time, the engine was on in order to receive the emails. After testing, we realized that the engine could extract the images even in the pdf files and could create the corresponding HTML files without error.
- Click here to see the corresponding commit.

4.5 Advantages

Although there may be other solutions to solve the task, the use of this engine possesses several advantages compared to other approaches. One of these advantages is the independence of the engine. For example, the email server may be needed to change, so the only this that the engine requires is the new configuration. Also, another problem that our engine will not face is the loss of archives. Suppose, in the future, we decide to utilize another email service. Therefore, because we have stored all id cards in a separate table, we will not lose our archive of users' IDs. In contrast, if we only rely on an approach that provides a direct link, after changing the email service, we will no longer have access to the archive.