# **CSE327 Project Report**

Project Name: Read

Section: 1

Date of Submission: 04/06/2020

Submitted to: Nabeel Mohammed

Semester: Spring 2020

Group Members:

|  |  |  |
| --- | --- | --- |
| Name | ID | Email |
| Mehran Siddiqui | 1822107642 | mehran.siddiqui@northsouth.edu |
| Humayan Kabir | 1511089042 | humayan.kabir@northsouth.edu |
| Ishrat Jahan Ananya | 1631636042 | ishrat.jahan16@northsouth.edu |
| Aiaj Uddin Bhuiyan | 1621696042 | aiaj.bhuiyan@northsouth.edu |

# Introduction

Read is a web application for teachers and students primarily focused on document sharing. Teachers can open virtual classes which students can enroll into, providing an easy way to share documents to all students of a class. Viewing documents comes with two additional features: facial recognition and eye tracking. Students' identities are authenticated using facial recognition while eye tracking is used to ensure that students are looking at documents. Teachers are also able to look at statistics based on time spent viewing documents for each class.

In creating this application we made use of a number of different tools and technologies. Django was used as the backend framework, while the frontend was created using html, css, javascript and bootstrap. An open source facial recognition library which makes use of dlib, openCV and deep learning allowed us to integrate a modern method of identity verification[[1]](#footnote-0). WebGazer, an eye-tracking library entirely written in javascript was easily embedded into the frontend to detect eye gaze positions[[2]](#footnote-1). Finally, an API consistent with the principles of REST was developed using Django's Rest Framework[[3]](#footnote-2).

# Administrative Information

All information about group members is provided on the front page.

The project repository can be found [here](https://bitbucket.org/nabeel_mohammed/spring2020.cse327.1.6/src/master/).

# Project Status

Use cases which were proposed and completed:

For teachers

1. Create classes for students to join
2. Approve, decline student requests to join classes. Remove enrolled students from class.
3. Upload documents to a class
4. Validate students' identities using facial recognition when viewing files
5. Ensure students' are actually looking at documents using eye tracking
6. View statistics based on students and documents for each class

For students:

1. Join classes available from different teachers
2. View documents uploaded in each class
3. Get notifications about joining classes, time spent on viewing specific files
4. Upload photo for facial recognition

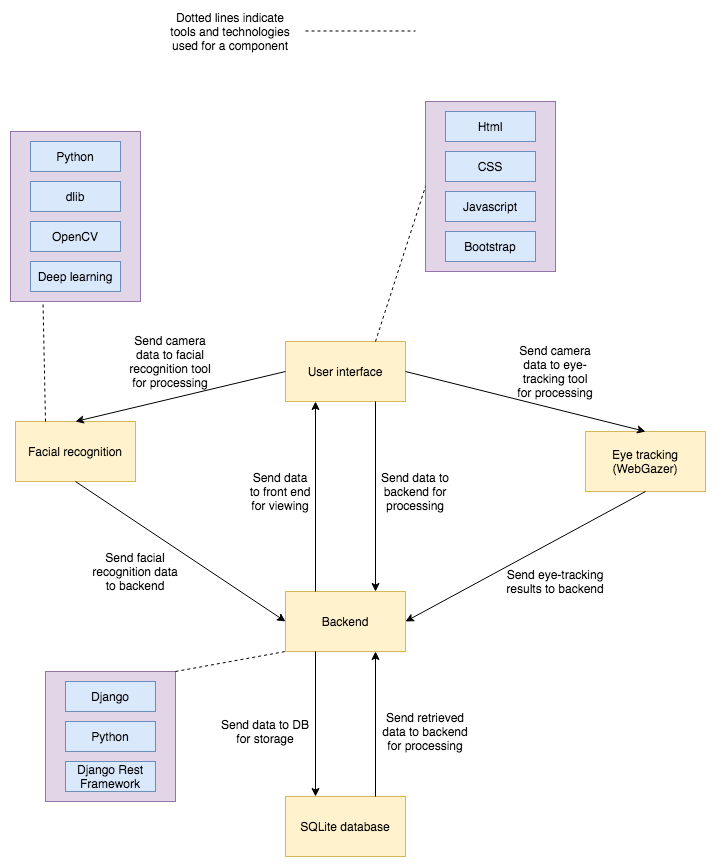
For both:

1. Use a google account to register and log in to the web application.

Use case which was proposed but not completed:

1. Allow teachers and students to use a mobile application with all the features as in the web application.

# Overall Final System Diagram

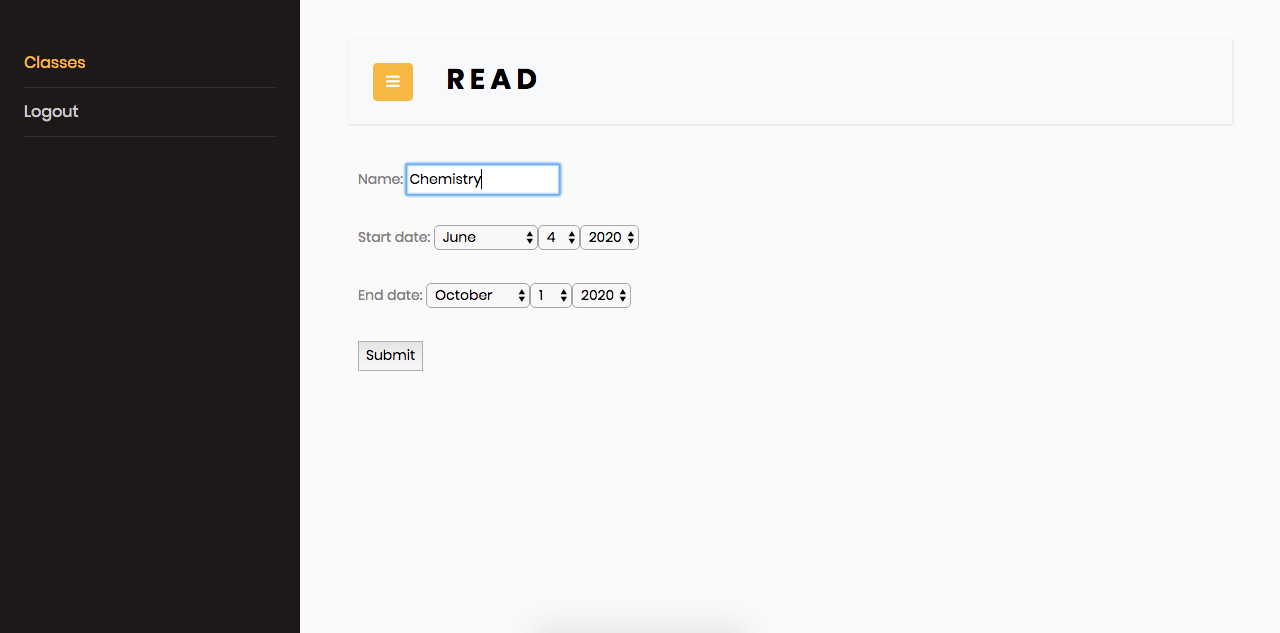


# Details of Different Platforms

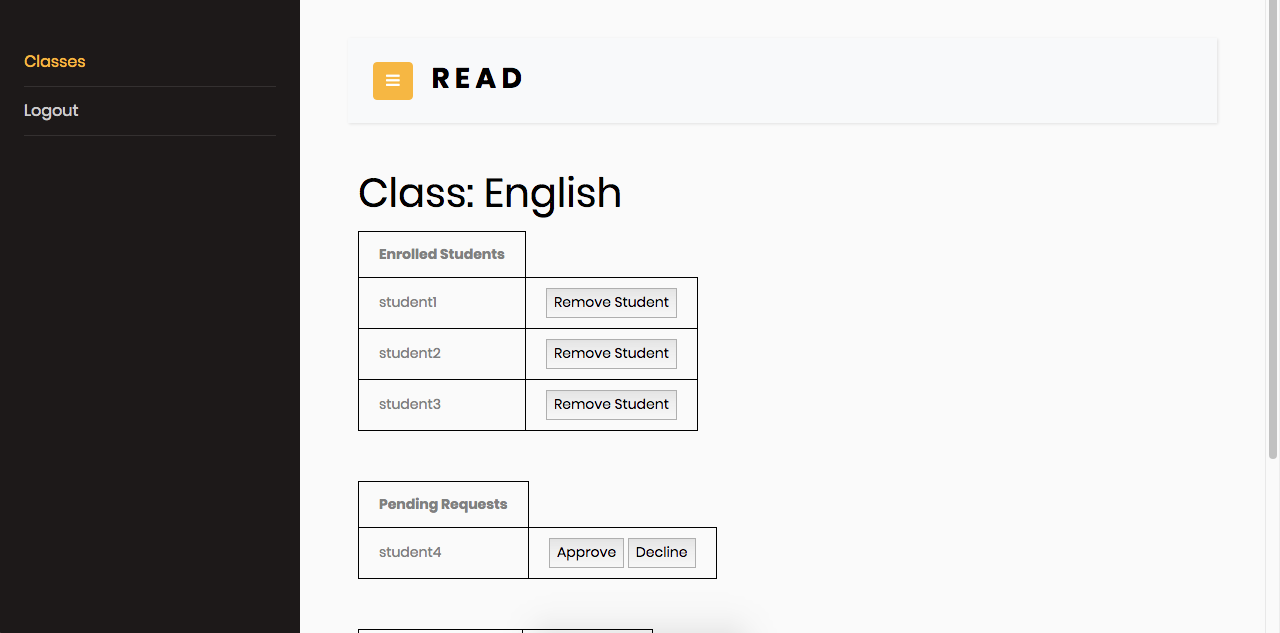
We have only created our product as a web application. All screenshots are from this web application.

For teachers:

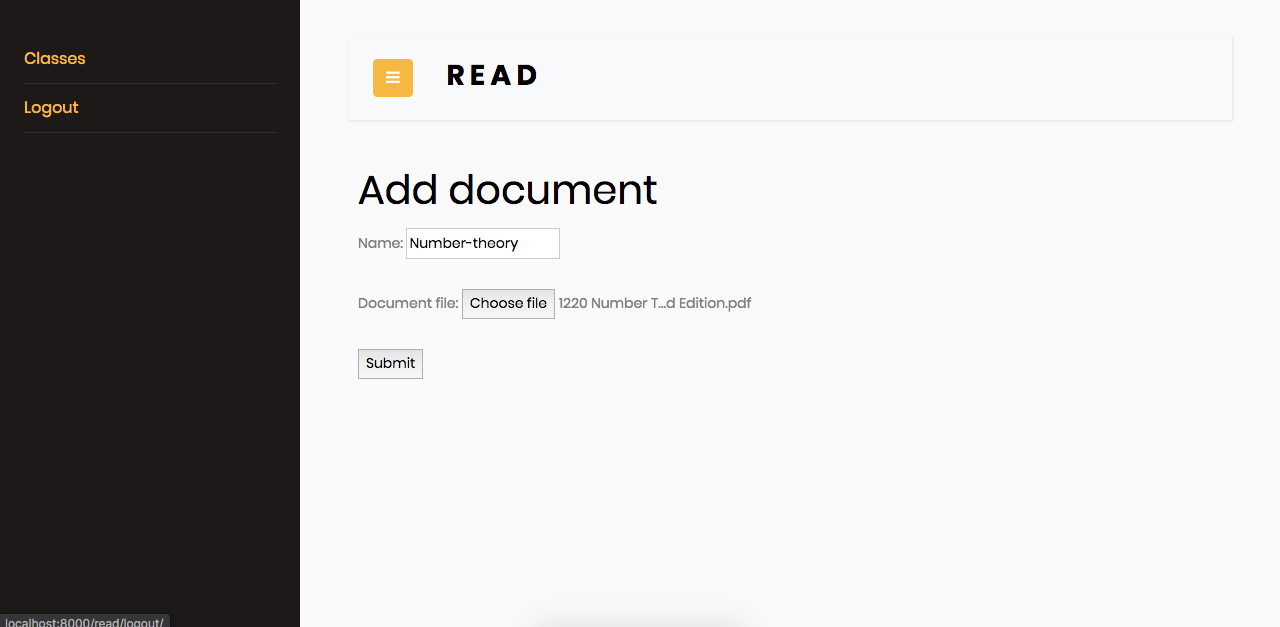
1. Use case: create classes for students to join  
   Description: teachers can create classes by providing a name for the class as well as the start and end dates. The class name has to be unique and the end date has to be after the start date. After the class creation, it becomes available for all students to join.



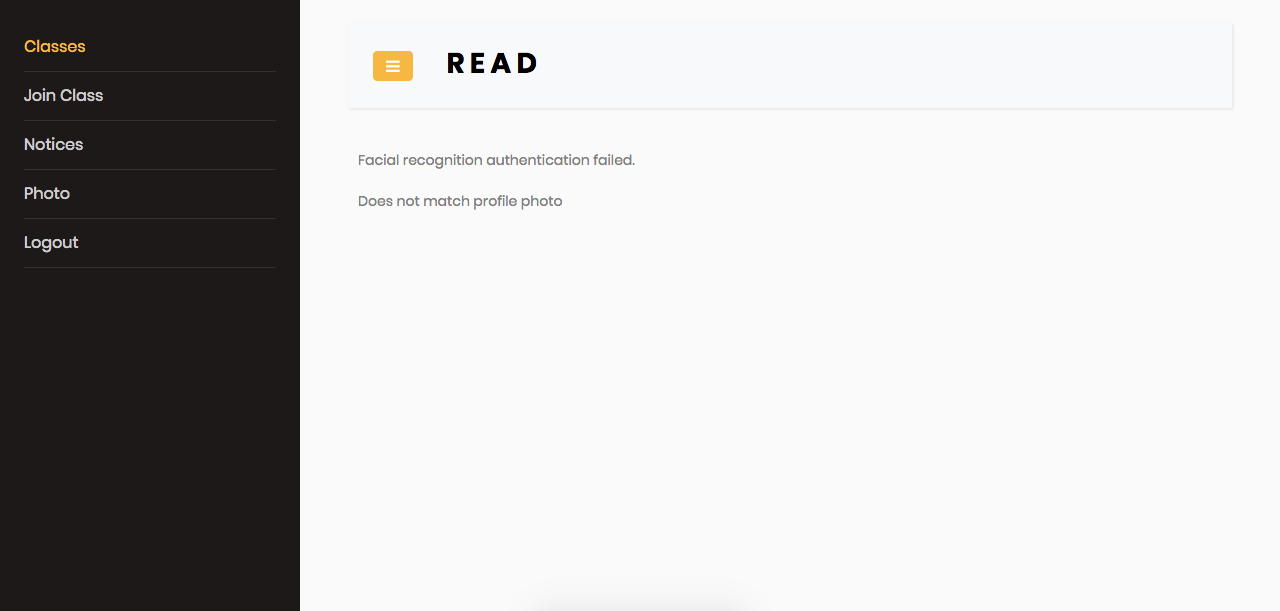
1. Use case: approve, decline student requests to join classes. Remove enrolled students from class.  
   Description: for each class the teacher can view all requests from students to join the class. The teacher can approve or decline this request. Existing students can also be removed from the class. Documents available to the student become unavailable upon removal from the class.

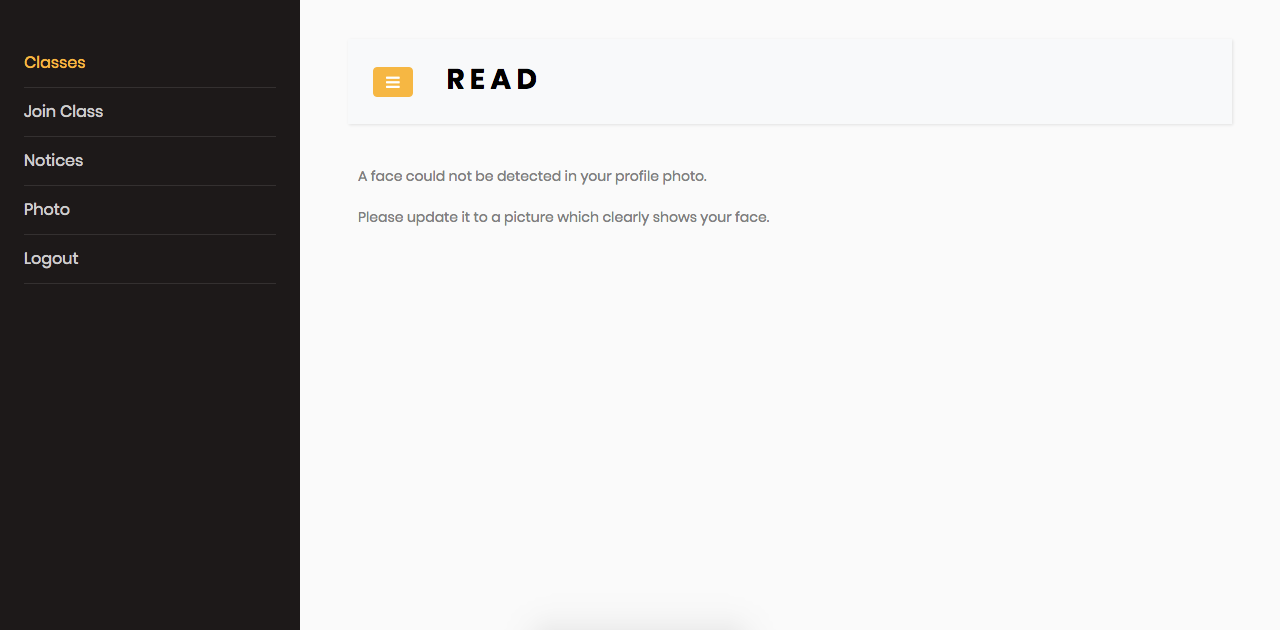


1. Use case: upload documents to a class  
   Description: teachers can upload documents to each of their classes. All document names within the same class must be unique and adhere to some constraints such as having no spaces.

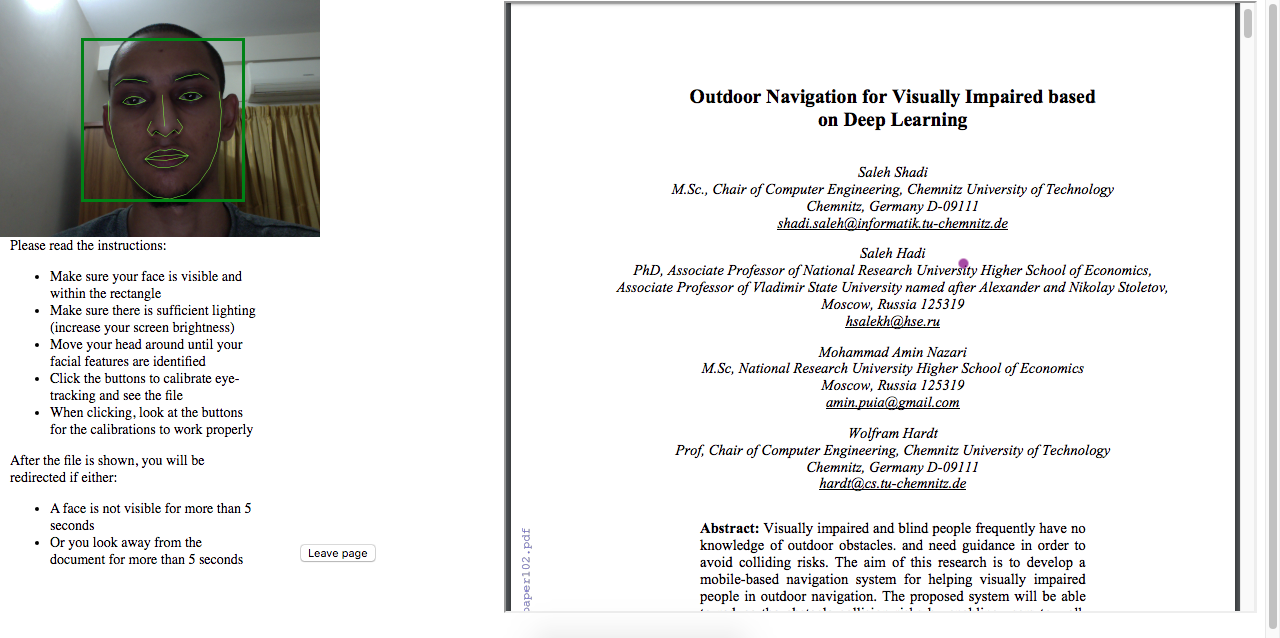


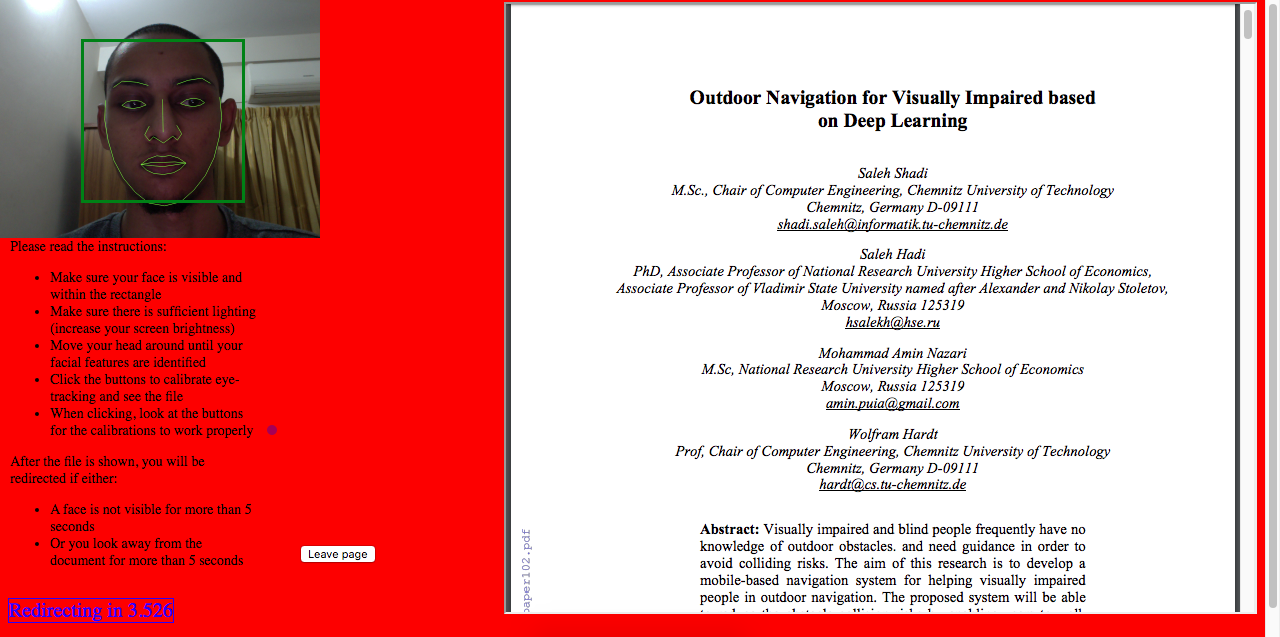
1. Use case: validate students' identities using facial recognition when viewing files  
   Description: when a student tries to view a file, facial recognition is initiated. If a photo is not uploaded, or a face is not detectable within the photo, a relevant message is displayed as shown in the two screenshots below. If the facial recognition is successful, students are directed to the page where they can view files.





1. Use case: ensure students' are actually looking at documents using eye tracking.  
   Description: after facial recognition, students are redirected to the page where they can view the file. A purple dot on the page indicates the position of the user’s gaze. The user can read the document which opens up to the right side of the page. If the user’s gaze goes out of the area of the document, the background turns red and a timer is shown in the bottom left (second screenshot below). Once this timer reaches 0, the user is redirected away from this page, meaning if they want to view this file, they would have to go through the facial recognition process again.





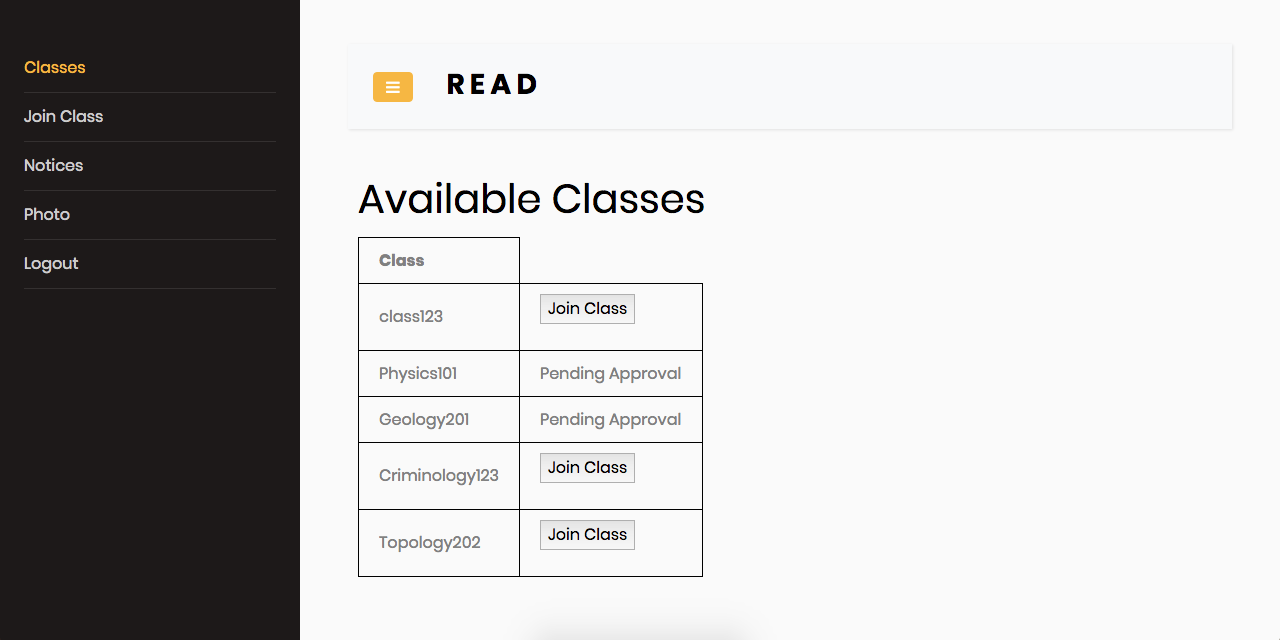
1. Use case: view statistics based on students and documents for each class  
   Description: teachers can view two different graphs for each class. The first graph shows a plot of the total time spent on all documents by each student. The second graph shows the total time spent on each document by all students.



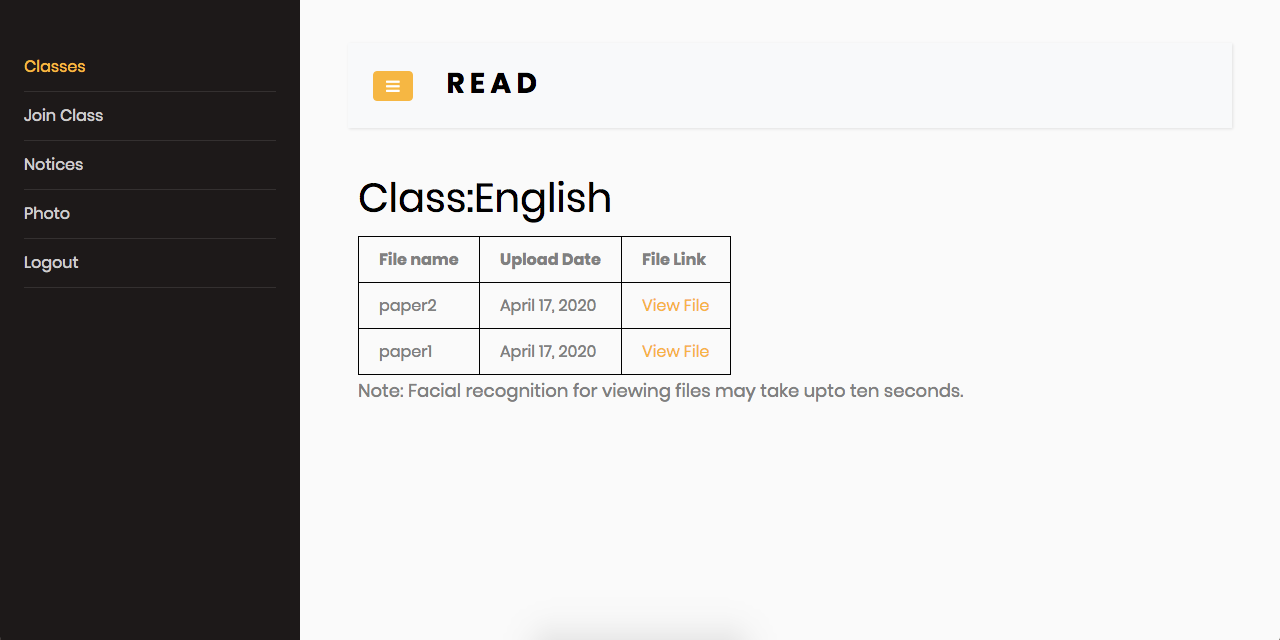


For students:

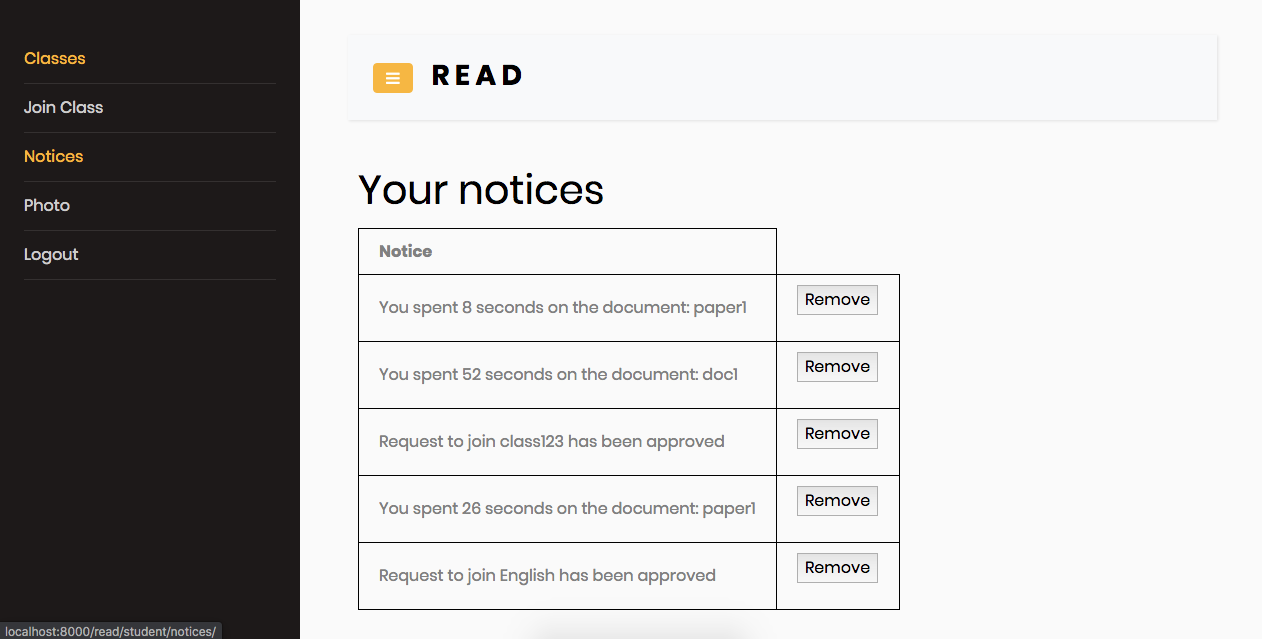
1. Use case: join classes available from different teachers.   
   Description: this is a list of the classes that are available and in which the student is not enrolled. Upon clicking the join class button, the status changes to pending, which the teacher of the class can choose to approve or decline.



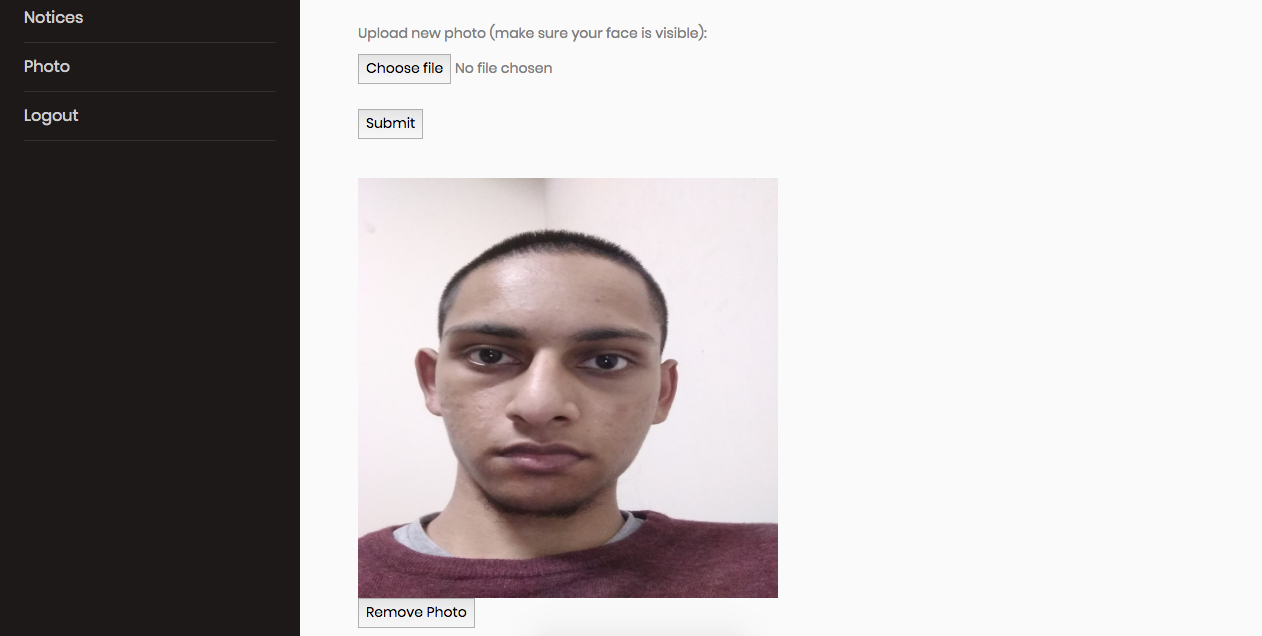
1. Use case: view documents uploaded in each class  
   Description: a list of all documents available within a particular class. Clicking the link on the document initiates facial recognition and allows the student to view the file.



1. Use case: get notifications about joining classes, time spent on viewing specific files.  
   Description: students get a notice when their request to join a class is approved or declined. After viewing a document, the time spent on that document is also shown as a notice.

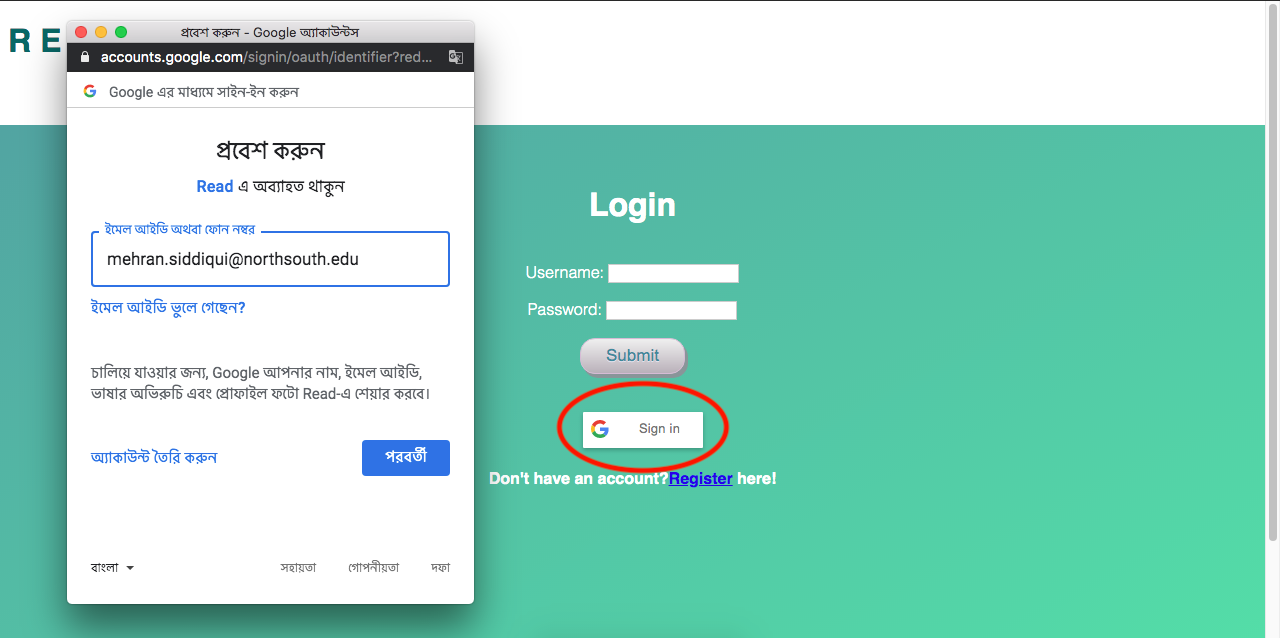


1. Use case: upload photo for facial recognition.  
   Description: students can upload a photo, which is matched with the face detected during the facial recognition process.



For students and teachers:

1. Use case: use a google account to register and log in to the web application.  
   Description: users can choose to register with their google accounts in order to create accounts. Some bits of information, such as whether a user wants to register as a student or a teacher, cannot be determined from the user’s google account. Consequently, an additional page is shown (second screenshot below) that asks for these additional fields to be filled in. Subsequently logging in either with a google account or without, allows the user to be directed to the corresponding homepage depending on the user type.



# 

# API Details

We created an API that allows querying each model class of our database. This allows adding, deleting and modifying any entry of any model class.

For a list of all entries of a particular database table (or model), the endpoint follows the pattern /read/api/table\_name/. For a specific entry within a table, the pattern /read/api/table\_name/entry\_primary\_key/ is followed. For example, /read/api/students/ provides a list of all students, while /read/api/students/4/ provides information about the student with primary key equal to 4.

In creating this API, we made use of Django’s Rest Framework. This allowed us to bypass some tedious tasks, such as having to write the code for querying the database for each endpoint and encoding/decoding between python native data types and JSON. Moreover, this framework provides a web interface for the API that can be easily browsed.

Interacting with this API programmatically or using command line tools works just as well, however for the purpose of exploring the API we recommend using this interface. After running our web application, simply going to any of the endpoint urls in a browser brings up the interface.

## Documentation

For the full documentation of this API, see [this file](https://docs.google.com/spreadsheets/d/1gkRTmgck8wv3YNikJeppmJ9dwG6kAXArVH9dbBbo6e0/edit?usp=sharing).

## Relevant files

|  |  |
| --- | --- |
| Link | Relevant Section |
| [API Logic](https://bitbucket.org/nabeel_mohammed/spring2020.cse327.1.6/src/master/web_app/mysite/read/api_views.py) | Whole file |
| [Serializers](https://bitbucket.org/nabeel_mohammed/spring2020.cse327.1.6/src/master/web_app/mysite/read/serializers.py) | Whole file |
| [Endpoint Paths](https://bitbucket.org/nabeel_mohammed/spring2020.cse327.1.6/src/master/web_app/mysite/read/urls.py) | Lines 36-59 |

## List of Endpoints

There are 16 endpoints for our API. The following is an exhaustive list:

|  |
| --- |
| /read/api/users/ |
| /read/api/users/<username>/ |
| /read/api/students/ |
| /read/api/students/<student\_primary\_key>/ |
| /read/api/teachers/ |
| /read/api/teacher/<teacher\_primary\_key>/ |
| /read/api/classrooms/ |
| /read/api/classroom/<classroom\_primary\_key>/ |
| /read/api/documents/ |
| /read/api/documents/<document\_primary\_key>/ |
| /read/api/enrolled\_ins/ |
| /read/api/enrolled\_ins/<enrolled\_in\_primary\_key>/ |
| /read/api/student\_documents/ |
| /read/api/student\_documents/<enrolled\_in\_primary\_key>/ |
| /read/api/student\_notices/ |
| /read/api/student\_notices/<student\_notice\_primary\_key>/ |

1. "ageitgey/face\_recognition: The world's ...." <https://github.com/ageitgey/face_recognition>. Accessed 2 Jun. 2020. [↑](#footnote-ref-0)
2. "WebGazer.js - Brown University." <https://webgazer.cs.brown.edu/>. Accessed 2 Jun. 2020. [↑](#footnote-ref-1)
3. "Django REST Framework." <https://www.django-rest-framework.org/>. Accessed 2 Jun. 2020. [↑](#footnote-ref-2)