



Bilkent University

Department of Computer Engineering

---

# Senior Design Project

*Project short-name: Here!*

## Final Report

Mert Aslan, Rahmiye B şra B y kgebiz, Hakkı Burak Okumuş, Ozan Aydın, Y ce Hasan Kılı 

Supervisor: Abdullah Erc ment   ek

Jury Members:    dem G nd z Demir and Can Alkan

Final Report  
Apr 30, 2021

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS492.

# Contents

<b>Introduction</b>	<b>4</b>
<b>Requirements Details</b>	<b>4</b>
Functional Requirements	4
Non-functional Requirements	5
Security	5
Pseudo Requirements	6
<b>Final Architecture and Design Details</b>	<b>6</b>
<b>Development/Implementation Details</b>	<b>8</b>
<b>Testing Details</b>	<b>10</b>
<b>Maintenance Plan and Details</b>	<b>12</b>
<b>Other Project Elements</b>	<b>12</b>
Consideration of Various Factors in Engineering Design	12
Ethics and Professional Responsibilities	14
Judgements and Impacts to Various Contexts	14
Teamwork Details	15
Contributing and functioning effectively on the team	15
Helping creating a collaborative and inclusive environment	15
Taking lead role and sharing leadership on the team	15
Meeting objectives	15
New Knowledge Acquired and Applied	15
<b>Conclusion and Future Work</b>	<b>16</b>
<b>Glossary</b>	<b>16</b>
<b>References</b>	<b>16</b>
<b>User Manual</b>	<b>18</b>
<b>Landing Page</b>	<b>18</b>
Register and Login	18
<b>Main Page</b>	<b>19</b>
Join Session	19
Display Notes	20
Add Courses	21
Assign Student to Courses	22
<b>Conference Page</b>	<b>22</b>
Conference Utilities	23
Chat Panel	23
Shared Slide	24
Note Taking	25
TA	26
<b>Analytics Page</b>	<b>26</b>

# Final Report

*Project Short-Name: Here!*

## 1 Introduction

The COVID-19 pandemic that struck the world in early 2020 led humanity to search for alternate solutions to everyday tasks. It affected people to the point that these tasks which would have been quite trivial in the past, now require immense care to complete. Along with these safety measures came the exercise of social distancing, which led to a complete overhaul of human to human interaction. We now communicate mostly online, with little to none face to face interaction. Video conferencing applications, such as Zoom, Google Meet or Microsoft Teams are much more prevalent than ever, due to this unprecedented circumstance. Even though this new type of communication is necessary to prevent the spread of the disease, it certainly has its drawbacks.

As students, during our time being educated remotely, we have experienced some problems that affected both the instructors and the students. It is evident that an online education setting has much less room for student-instructor interaction compared to a classroom setting. This mainly has to do with the fact that video conferencing applications such as Zoom or Google Meet are not specialized to be used as an education medium. Many valuable information, such as the attention of the students to the lectures, that were constantly gathered by instructors in classrooms are lost in online education due to the fact that these conferencing mediums are simply not specialized enough to gather such information. Our experiences show that the loss of such information leads to a complete detachment between the instructors and students, which overall diminishes the effectiveness of online education. Our application aims to overcome this problem, by providing a solution using machine learning and computer vision algorithms that would increase the amount of knowledge the instructors receive about their students both in real time and right after lectures.

Students also lose their ability to effectively follow the lessons, as these video conferencing mediums do not provide them any means to enhance their learning process. They are now essentially “participants” rather than “students” that do not have the essential tools they once had in their classrooms. This unfortunate situation distances them from the learning process itself. We also aim to solve this problem, by providing a platform specialized for the needs of the students in an online education environment.

In this report, the decisions that we have made throughout the previous reports will be finalized. The final form of the requirements, architecture and implementation will be given and changes will be discussed. We will mention the testing and the future maintainability of the system, then talk about the teamwork and the future of our project.

## 2 Requirements Details

### 2.1 Functional Requirements

- Users are able to give video input to the application using any webcam.
- Users are able to give audio input to the application using any microphone.
- Users are able to register to the application as an instructor or a student.
- Instructors are able to register a course to the system.

- Instructors are able to assign students to a course.
- Students are able to see their weekly schedules.
- Instructors are able to start a lecture.
- Students are able to join a lecture.
- Users are able to open/close their cameras and microphones during the lecture.
- Students are able to view the instructor's video and their shared screen in a lecture.
- Students are able to navigate in the shared slides (if any) independent from the instructor and synchronize back with the instructor when they want.
- Students are able to open a notepad on the side while listening to the lecture and view their notes after the lecture.
- Users are able to access the chat panel during the lecture.
- Users are able to view the participants list during the lecture.
- Instructors are able to mute/unmute students.
- Students are able to raise their hands physically to begin to speak.
- Instructors are able to receive notifications when students raise their hands.
- Instructors are able to receive statistics regarding the attention of the students to the lecture both in real time and after lecture.
- Students are able to receive notifications from the TA when they are distracted, using their phone, etc.

## 2.2 Non-functional Requirements

- Security  
The program ensures that video recordings of users will not be shared with any other 3rd party application. Users must accept the terms of service and privacy policy to use the program. Also the program does not store any kind of image or video, and stores sensitive information like passwords in a hashed, secured form.
- Usability  
The program is suitable for webcams with different resolutions. The program is also supported by different browsers. Unless the environment is too dark, the program will be able to recognize hands, head poses and cellphones very accurately.
- Performance  
The program utilizes microservices and queues to be able to communicate the results of the machine learning algorithms back to the user with minimum delay.
- Extensibility  
The program is developed as a web application at first but it must be extensible for desktop, Android and iOS environments.
- Scalability  
Our backend is deployed with GKE, which provides both horizontal and vertical scaling in various forms. Our load balancer is also scalable, meaning that our program is prepared to handle many requests and backend operations at the same

time without experiencing down time. Our WebRTC servers can handle 20 people on average in a single session with acceptable delay, but more people usually means more delay for everybody since we are using mesh network topology.

- Reliability

We have designed our microservices such that the dependency among them is minimum and one can function without the others most of the time. We are keeping at least 2 pods for each microservice alive any time so that in terms of a system failure, the other pod can take over. Our WebRTC servers can handle cases where a student is disconnected from the session, such that students can easily rejoin and continue the lecture.

## 2.3 Pseudo Requirements

- Backend implementation language is Python 3.8.
- The frontend is implemented using Angular, and written mainly in TypeScript.[6].
- To add real-time communication capabilities to the program, WebRTC framework is used [3].
- There are two databases, one PostgreSQL for the relational database tables and one Redis for the Celery operations.
- GitHub and Trello are used for collaborative work and issue tracking.
- OpenCV,dlib, keras and mediapipe libraries are used for motion tracking capability of the TA feature of our project.
- Darknet Convolutional Neural Network framework together with its YOLOv3 tool is used for object detection in order to enhance TAs tracking capabilities.
- Project is deployed using GKE with Docker-compose and Kubernetes is used for controlling docker containers.
- Locust and Jest frameworks are used for testing.

## 3 Final Architecture and Design Details

In the previous reports, our architecture was a monolith architecture where every system was extremely dependent on each other and scaling the system was only possible via vertical scaling. During the course of our implementation, we realized that this was not possible due to our need of running multiple machine learning models, which requires individual scaling. Since then, we have turned our architecture into a microservice-cloud native architecture where it is possible to scale individual components horizontally without taking resources away from other microservices.

Our client side, which is written with Angular, is deployed using Github Pages. Since we do not need to scale the client side, we handle some of our program logic here so that we can have a more light-weight backend.

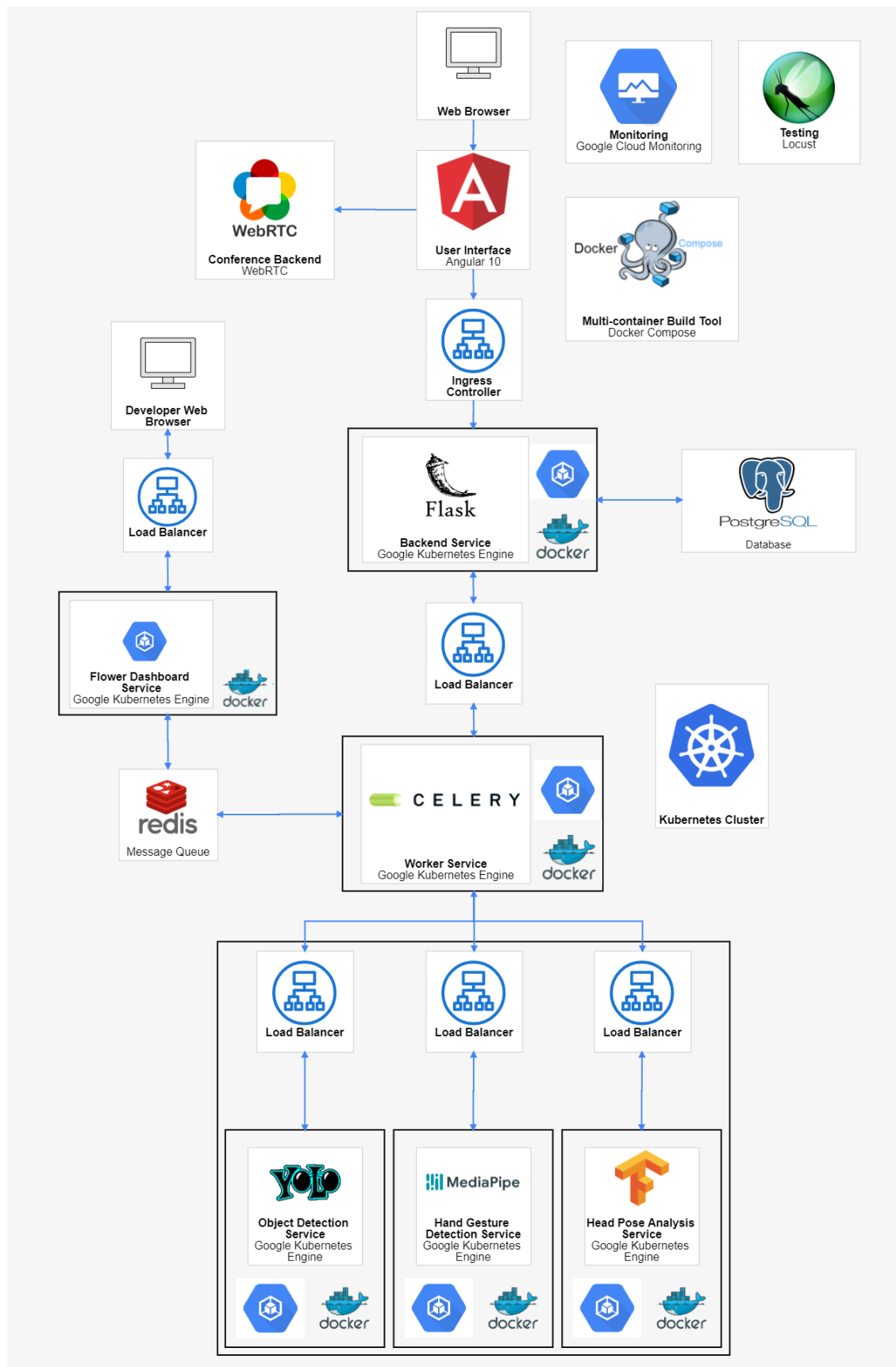
The services inside our Kubernetes cluster use an Ingress controller to modulate any incoming HTTP/S requests. Ingress is scalable and can create many load balancers to suit our needs. It also directs requests to services and thus, responsible for routing.

We have multiple services inside our GKE deployment. Clients send their requests to our "web" microservice, which is our main application written in Python. Inside this web application, we are running Celery, a task queue that allows us to handle machine learning requests without making the client wait for the results. Celery uses workers to actually

execute tasks that it was given, so one of our services is the Celery worker, where we can scale up and down easily, depending on the network traffic. We also have a Flower Dashboard service where we can maintain and monitor our celery workers, queues and tasks. Another service that we have is the Redis service, which serves as a message broker for Celery. Our Redis service stores the incoming tasks and deals them to the workers accordingly, while also keeping the results of each completed task.

Apart from the services mentioned above, each of our machine learning algorithms are also services themselves, which was a crucial design choice in making our project work. This choice allowed us to scale our machine learning models individually, which was needed as both of these models take different times to work and require different amounts of CPU and memory, so it only makes sense that they are separated. Our "web" service receives images periodically that we need to feed into our machine learning models, so inside our web service we call our machine learning services asynchronously using Celery. After a result is calculated, it is communicated back to our web service so we can make necessary calculations.

## 4 Development/Implementation Details



**Figure 1:** Deployment Diagram

- **Mediapipe:** We used Google's new open-source "cross-platform, customizable ML solution [4]" in our project in order to do hand detection on images. It is a very lightweight and customizable technology that allows really fast results in terms of hand recognition and many others.
- **WebRTC:** At the start of the project, we thought WebRTC is a simple API that allows people to create real-time communication applications. During the project, we realized that WebRTC is much more complicated and required in-depth knowledge about P2P connections, socket programming, network topologies such as SFU and different servers like ICE, STUN etc.
- **PostgreSQL:** For our relational database management, we decided to go with PostgreSQL since it is a reliable RDBMS that there is no uptime or data integrity issues, it has a wide range of data types which gives us flexibility and it has a good performance in terms of speed.
- **AWS:** At the design state of our project, we decided to use AWS EC2 for our backend server, AWS RDS with MySQL for our relational database storage and AWS S3 to store relatively big files such as pdfs. However, during the development process, we observed that we were running out of the quota provided to Free Tier users and we found out that AWS will be too costly for us. Therefore, we decided to move our backend servers to another PaaS provider.
- **Google Kubernetes Engine(GKE):** We are running 3 different machine learning models constantly, which is a really heavy operation. We experimented with it locally and some free tier Platform-as-a-Service providers like Heroku but the computational load was too much and the server couldn't keep up. As our product needs to serve multiple users with long-running background tasks, it was inevitable to have a design that made scalability the priority. Therefore, we decided to turn each of these machine learning models into a microservice and deploy them to GKE, with the help of Docker. Using Kubernetes allowed us great flexibility and opportunity to scale, as we could scale our models by creating or removing pods, distribute CPU usage among them, distribute the requests coming in and going out and monitoring our every step.
- **Docker:** As we have discussed in the GKE section, each of our machine learning analyzers are divided into microservices. Each of these microservices have their own Dockerfiles that have the build instructions needed to run each one of them. This way, we were able to separate functionalities into different containers that work independently and that have a single responsibility. This enabled us to develop our services according to the S.O.L.I.D principles.
- **Kubernetes:** Kubernetes is a container orchestration tool, which enables full control over the docker containers using clusters, nodes and pods. We currently have a Kubernetes cluster deployed to GKE, which we can manage through the Kubernetes command line tool "kubectl". Our cluster has a node pool consisting of 3 nodes, and each node has a differing number of pods that are running the docker images of the microservices. The number of the pods are automatically scaled up or down according to the current CPU and memory usage of the pods, as well as the number of requests the external cluster load balancer receives each second.
- **Flask:** Flask is a micro web framework written in Python. We are using the Flask framework and many of its different libraries in our backend to provide a RESTful API to the consumers of our services.
- **Celery:** In our project, we have to constantly run machine learning models, from which we have to get results and communicate these results to the front end. The



problem is, machine learning models each run in different periods and lengths, and the front-end cannot wait for the results as that means delaying the whole system. This is where a task queue like Celery comes in. With Celery, we can queue different tasks into different queues, assign differently configured workers to these queues so that we can efficiently run our machine learning algorithms in the background.

- **Locust:** Locust is a load tester that is written in Python. We used Locust for our most crucial tests like checking how many requests our microservices can handle in a second and with how much latency, the performance of our load balancer and of course the performance of our Celery queues and workers.
- **Flower:** Flower is an open-source real-time Celery monitoring tool. Flower became a crucial tool for us as it allows us to see how many workers and queues we have, how many tasks are being processed and executed, how many of them failed, and much more. Using Flower enabled us to tune and optimize our Celery usage.
- **Redis:** Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker [7].” We are using the message broker functionality of Redis in order to store the results produced by our machine learning microservices, which are later on queried by other services. We are able to prevent global system lock-ups as we are asynchronously analyzing the images and storing the results on Redis.
- **Docker-Compose:** In order for our product to function, there needs to be 3 separate services running concurrently, which are Flask, Celery and Redis. We have also added an optional fourth one, which is Flower, to be able to monitor the status of the Celery workers and the Redis message queue. All of these services have their images built by separate Dockerfiles, except Redis, which has its own image in Docker-Hub. Docker-Compose helped us tremendously to be able to manage the dependencies of each of these images, build and run them concurrently.
- **Kompose:** Kompose is a tool that converts a docker-compose specification file into separate kubernetes-ready resources. We used Kompose in order to generate Kubernetes deployment and service resources from the aforementioned Docker-Compose file that we can use to run on Google Kubernetes Engine.

## 5 Testing Details

We have used two main frameworks for two very important areas of testing. First, in order to test the front end functionality, which is written mostly in TypeScript, we have used the Jest framework. We ran our Angular unit tests using jest-preset-angular package. We chose Jest because it is easy to use and it has powerful mocking capabilities and built-in code coverage generator. After configuration, it takes only one line to run all tests on the front-end side.

Secondly, in order to test the limits of our backend, we have used Locust as our load testing framework. As we plan on serving multiple users concurrently, our backend services should be able to adapt to the amount of traffic it receives. Locust is a very flexible Python based load tester where we can simulate any number of users and their requests in order to check how many requests our system can handle, and the amount of latency for every request. It allows us to run a custom script that simulates the actions of a user or the system, and check whether the server can handle multitudes of that specific request. Locust was crucial because we were using Celery queues and needed to test the capabilities of the workers and how frequently they can receive tasks before

crashing. Locust enabled us to tune the performance of our workers. It also enabled us to check average response times, as for hand gesture recognition specifically, fast response times are crucial for our program. Below, you can find sample results from our load test performed on the hand gesture recognition microservice.



Figure 2: Locust Load Test Result for Hand Recognition Endpoint

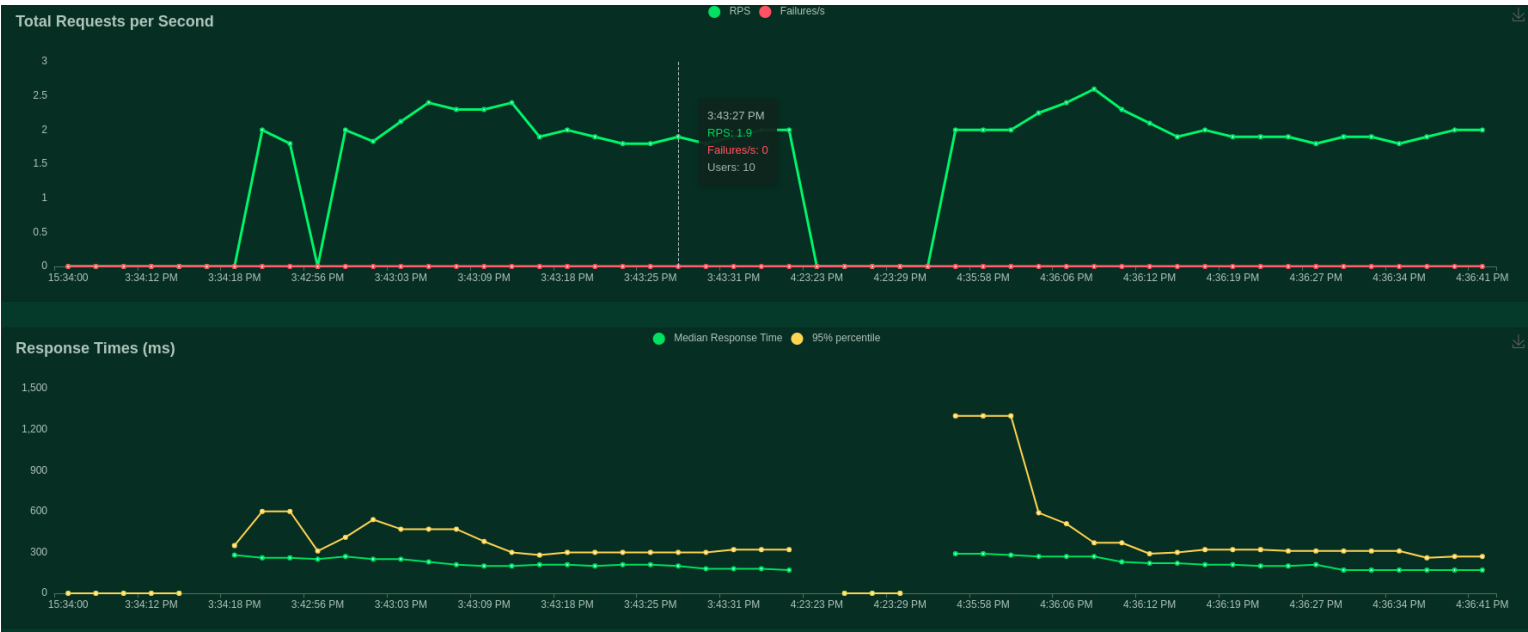
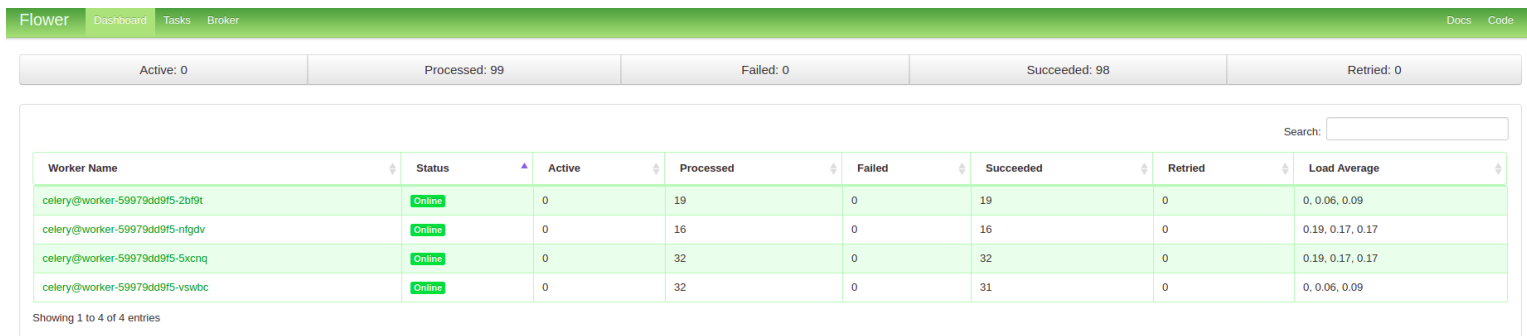


Figure 3: RPS and Response Time Charts

The hand gesture recognition microservice was able to handle 2 requests per second without any failures. 95th percentile of these requests had a latency of 270 ms, which is highly tolerable for our purposes. According to our test results, we are confident that our product is able to scale to serve higher loads when necessary.



**Figure 4:** Flower Dashboard

We are also able to monitor the status of the Celery workers through the Flower dashboard. As it can be seen from the dashboard, all of the 97 requests sent from Locust were successfully completed by the 4 Celery workers running on GKE. The extra 2 processes are leftover from an earlier testing done. The result of these completed tasks are kept on the selected message broker, which is Redis, until a query is made to learn its results.

## 6 Maintenance Plan and Details

Since our backend is deployed using GKE in Google Cloud Platform, further maintenance is established through GCP. All the errors and warnings can be monitored through GCP logs, so in order to maintain our project, thorough monitoring is needed. This is especially important as currently, due to money constraints, we are not using the Autopilot cluster for our project, which enables auto scaling based on the server load. Since we are not using Autopilot, we have to carefully monitor our server load and if needed, replicate pods and distribute power among them so that we do not experience downtime. For better maintenance, we might consider switching on Autopilot mode as well.

Maintenance on our Redis servers and databases are also crucial for the continuation of our system. We are using a Redis task broker that is deployed within GKE for Celery, which has a limited amount of storage of 2GB's. If our celery tasks are not maintained properly, then tasks begin to cluster inside our Redis storage, which can mean downtime and memory failure. So, we have to use Flower efficiently to keep track of Celery tasks, and carefully monitor Redis logs. We are also using a PostgreSQL database for our main database, which is known as a bit less efficient but much more reliable database, which is an important factor for the maintenance of our application. Database reliability is important as any downtime experienced in the database really hinders user experience. Since our database is deployed on Heroku, we can use the Heroku logs for detecting any possible problems.

## 7 Other Project Elements

### 7.1 Consideration of Various Factors in Engineering Design

#### Public Health

Public health is the motivation of our project. Due to the ongoing pandemic, it is crucial that the students remain in their homes, as being in a real classroom is a huge health hazard. Our project ensures that the students remain in isolation while trying to provide an education tool that resembles the feeling of an actual classroom.

## Public Safety

Due to the ongoing pandemic, it is essential, regarding public safety, to stay isolated. Due to this factor, our team cannot get together in the same environment as often as we would like, which is making the development process harder in terms of cooperation and teamwork.

## Public Welfare

A country's welfare state affects its average internet connection speeds and accessibility of better electronic devices. Both affect user experience quality of our program, as our program relies on visual data communication, which can be an expensive process regarding internet connection. On the other hand, currently our program is free to use, so it has no real effect on public welfare.

## Technological Factors

Technology is constantly developing, and this is an important thing to consider for our project. We need to keep track of this development and implement it to our project so that our work is not obsolete. We also need to look out for similar products, and develop our application accordingly so that our product can add something to the table.

## Cultural Factors

Clothings and accessories may differ from culture to culture. These differences affect motion tracking and recognition of facial expressions.

## Social Factors

Users need to interact with the program using a language they are familiar with. In order to make our program available world-wide, we need to consider translating our program to different languages. Our program will be English as default, and language might be an issue since approximately 20% of the Earth's population can speak English.

## Global Factors

Global factors do not have a significant impact on the design or analysis parts of the project.

## Environmental Factors

Environmental factors do not have a significant impact on the design or analysis parts of the project.

	Effect level	Effect
Public health	10	Public health crisis motivating our project idea
Public safety	5	Hindering the cooperation and teamwork aspect of development due to isolation.
Public welfare	5	Change in user experience quality based on purchasing power

Technological factors	10	Change in design due to support for technologies we use and existence of similar applications
Cultural factors	4	Change in accuracy of image processing because of cultural differences on appearance
Social factors	6	Language barriers between users and the program
Global factors	0	-
Environmental factors	0	-

**Table 1:** Factors that can affect analysis and design

## 7.2 Ethics and Professional Responsibilities

Since we are dealing with live camera feed, it is important for us to consider the ethical side of our program. The images we collect from the students are only used to gather relevant information about a student's hands or if they are distracted, and they are discarded immediately after being fed into the models. It is very important that our program does not store the images taken from the live feeds of the students in order to maintain privacy. We also use a hashing algorithm to safely store the passwords, and we as system admins do not have access to user passwords.

Our professional responsibilities include the professionalism within our team. Project management ethics is important while managing projects. In order to maintain the professionalism within the team, we followed PMI's (Project Management Institute) code of ethics and Professional Conduct document. According to PMI, "Ethics is about making the best possible decisions concerning people, resources and the environment. Ethical choices diminish risk, advance positive results, increase trust, determine long term success and build reputations. Leadership is absolutely dependent on ethical choices" [8]. During our project, the leadership role has changed between the team members and each of the team members has adopted ideal and necessary behaviours in order to lead the project to success. Each of the team members has valued trust, honesty, respect and fairness to create and maintain the harmony and professionalism in the team.

## 7.3 Judgements and Impacts to Various Contexts

Judgement Description		
	Impact Level	Impact Description
Impact in Global Context	Medium	Since online education is a global thing now, our considerations went into how to make online education more feasible for any student in the world, which is why its impact level is medium.
Impact in Economic Context	Low	Since our application is free to use, the economic context was not a priority issue while making decisions about our program.
Impact in Environmental Context	Low	Our program does not include any

		extra technology that relies on environmental resources nor does it affect the environment any more than a standard application would.
Impact in Societal Context	High	Since we are using machine learning algorithms to understand if a student is distracted or not, we consider the societal impact of our decision so that the student does not feel bad or discouraged or even humiliated. For this, we decided to kindly warn the student and display anonymous statistics to the instructor instead of pointing fingers.

## 7.4 Teamwork Details

### 7.4.1 Contributing and functioning effectively on the team

For a project of this scale to function on such a tight schedule, teamwork is very important. As a group, we had frequent meetings in order to split the work among us, so that each member always has something to do. These feedback sessions were important as they allowed us to dynamically shift the workload to where it is most necessary at that time.

### 7.4.2 Helping creating a collaborative and inclusive environment

During the pandemic where group meetings are not really preferred, we used Discord, Trello and Github to create a collaborative environment.

### 7.4.3 Taking lead role and sharing leadership on the team

At different times, different people took initiative in our team. If a person is aware of the work that needs to be done and is capable of distributing that work among others equally, then that person is allowed to take a leadership role. This enhances inclusivity and helps create a workflow.

### 7.4.4 Meeting objectives

In order to meet our objectives, we had a meeting every week in order to assign new objectives and receive feedback for others. Trello and Github were a big help as visualizing and organizing tasks allow people to easily keep track of their objectives.

## 7.5 New Knowledge Acquired and Applied

During the project, our project design changed fundamentally in order to incorporate new and more suitable technologies into our project. Technologies like mediapipe, webRTC, AWS, GKE, Docker, Celery, Locust, Flower, Redis were new to us, and we had to gather information and more importantly experience about these. We acquired some knowledge from our other ongoing projects like our Cloud project, while for others we had to search the internet and read lots of documentation. We also had the privilege to talk to people who are currently working as a software engineer, so we learned a lot about Celery, GKE and Docker from them. Also some of our team members were not familiar with Angular, so

during the frontend development, all of our team members have acquired Angular framework knowledge.

## 8 Conclusion and Future Work

In conclusion, as fellow students, we believe that online education has been hard for both instructors and students as students tend to be more distracted, which not only affects their own performance but the morale of the instructors as well. In order to tackle this issue and some others that we believe are missing from the current tools that are being used in online education, we came up with Here!. Here! is using machine learning to detect distracted students, raised hands and cellphones in order to prevent distraction and make detailed analysis for the instructor so that they can improve with each lecture. On top of this it packs some new features that we as students feel necessary: simultaneous note taking and slide sharing, where a student can navigate through the slides without distracting the instructor. These features and possibly much more to come make Here! a more interactive and user friendly classroom environment that both students and instructors would want to use.

We believe Here! has much potential, and can definitely be enhanced and improved in the future. Currently, we are using the basic Google Cloud Platform tier which can only support limited amounts of microservices. By increasing our server power, we can add in new machine learning models to enhance our TA systems, and optimize our already existing models to perform much more efficiently. Besides slide sharing and note taking, we can come up with more problems that the majority of the students can relate to and interpret them in our program to be even more accessible to students. Overall, we are certain that our project has great potential to grow in the future.

## 9 Glossary

- **AWS:** Amazon Web Services, a package of web services including EC2 and RDS [1].
- **EC2:** Elastic Computing, Amazon based cloud server [2].
- **Flask:** Flask is a micro web framework written in Python [5].
- **Angular:** Angular is a TypeScript-based open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations [6].

## 10 References

- [1] "Cloud computing with AWS". [Online]. Available: <https://aws.amazon.com/what-is-aws/>. [Accessed: 29-Apr-2021].
- [2] "Amazon EC2". [Online]. Available: <https://aws.amazon.com/ec2/>. [Accessed: 29-Apr-2021].
- [3] "WebRTC", [Online] Available: <https://webrtc.org/getting-started/peer-connections> [Accessed Oct 10, 2020]
- [4] "mediapipe". [Online]. Available: <https://github.com/google/mediapipe/>. [Accessed:29-Apr-2021]

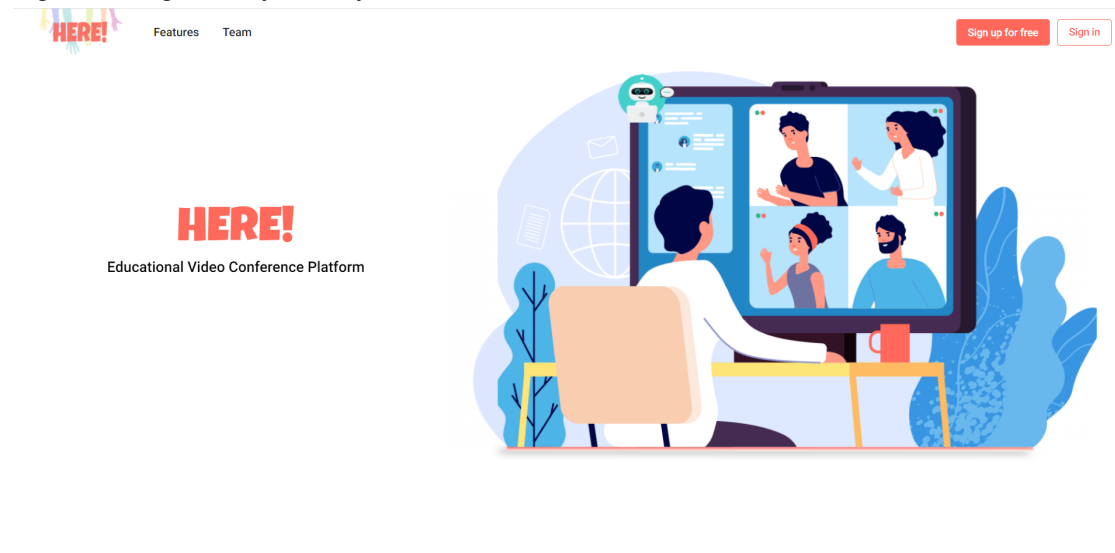
- [5] "Welcome to Flask¶," *Welcome to Flask - Flask Documentation (1.1.x)*. [Online]. Available: <https://flask.palletsprojects.com/en/1.1.x/>. [Accessed: 29-Apr-2021].
- [6] "Angular". [Online]. Available: <https://angular.io/>. [Accessed: 29-Apr-2021].
- [7] "Redis". [Online]. Available: <https://redis.io/>. [Accessed: 29-Apr-2021].
- [8] "Code of Ethics & Professional Conduct". [Online]. Available: <https://www.pmi.org/about/ethics/code>. [Accessed: 29-Apr-2021].



## 11 User Manual

### 11.1 Landing Page

Users will be greeted with the landing page of the application. Here, they can inform themselves about our project by clicking on the “Features” and “Team” buttons on the top left, register or login if they already have an account.



**Figure 5:** Landing Page

#### 11.1.1 Register and Login

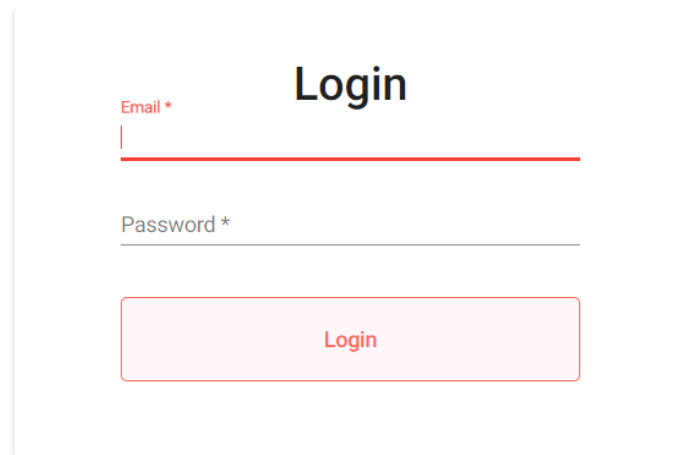
In order to register, users can click on the “Sign up for free” button, and in order to sign in, users can click on the “Sign in” button.



**Figure 6:** "Sign up" and "Sign In" buttons

The image shows a 'Register' dialog form. It has a title 'Register' at the top. Below the title are four input fields: 'Username \*', 'Email \*', 'User Type \*' (a dropdown menu), and 'Password \*'. At the bottom of the form is a red button with the text 'Register'.

**Figure 7:** Register Dialog



**Login**

Email \*

---

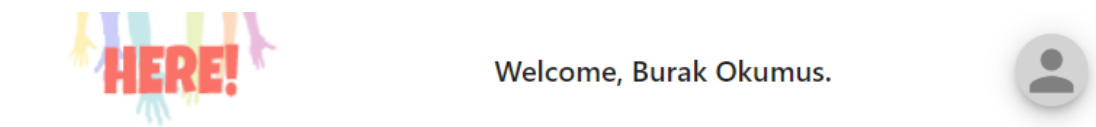
Password \*

---

Login

**Figure 8: Login Dialog**

Login and register screens are straight-forward, users can register by entering a user name, email, password and most importantly, a user type. If you are an instructor, you must click on the “Instructor” type, and if you are a student, you must click on the “Student” type. In order to log in, you must enter your email and password. After users are logged in, they will be redirected to the main page, with a message displayed on the toolbar that says “Welcome, User.”



**Figure 9: Toolbar after successful login**

## 11.2 Main Page

**My Weekly Schedule**



Hours	Monday	Tuesday	Wednesday	Thursday	Friday
8.30	CS 491				
9.30	CS 491				
10.30					CS 476
11.30					CS 476
13.30					
14.30					
15.30					
16.30					

**My Notes**



CS 491    notes saved on CS 491

CS 476    notes saved on CS 476

Note saved on 24/04/21

Note saved on 31/04/21

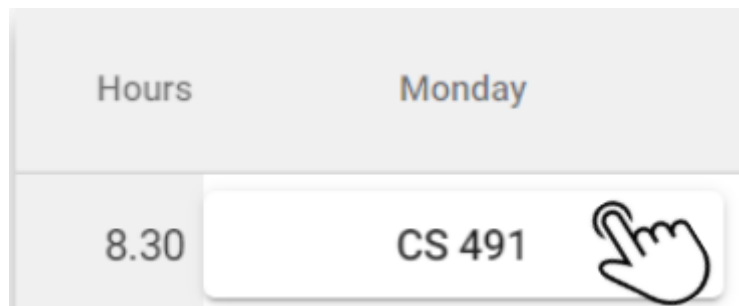



**Figure 10: Main Page**

### 11.2.1 Join Session

Main page includes a weekly schedule for both the instructors and students. Instructors are able to create courses, which will be displayed as buttons on the schedule. Instructors can add students to that course, and in doing so, the same button will be visible on the students calendar as well. Clicking on the buttons given below perform different tasks for different users. For instructors, for example, clicking on the button labeled CS491 will start a new

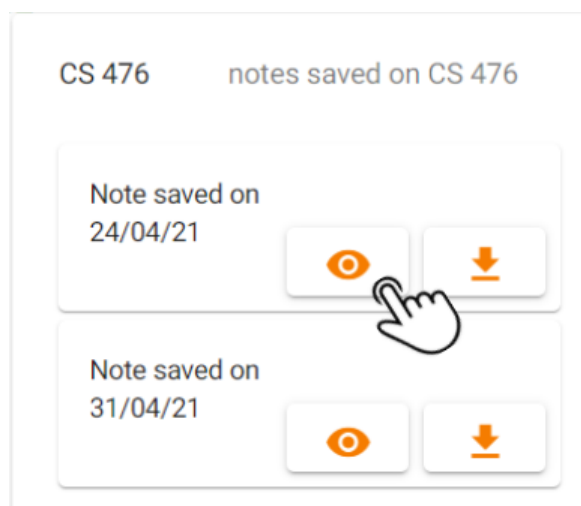
session for the course CS491. After the session is initialized, students will be able to click the same button on their schedules to join the session that is started by the instructor.



**Figure 11:** Button for joining a lecture session

### 11.2.2 Display Notes

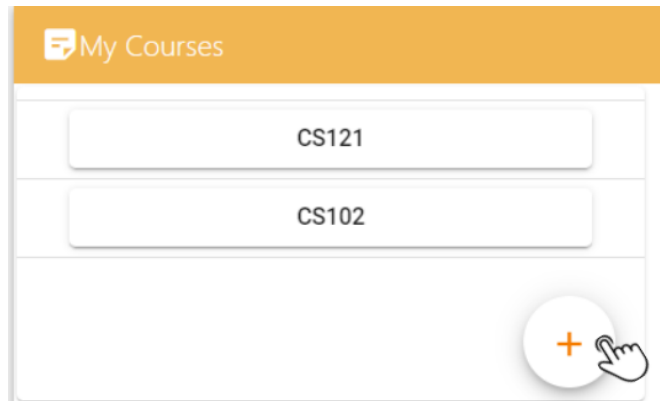
On the right side of the screen, students will see the “My Notes” section. This section contains notes that are saved by the student during lectures. Initially, students will see a list of their courses displayed there. Clicking on a course will open up another list, which displays all the notes taken in that course, labeled by their time. Users can click on the eye icon to view notes, or the arrow icon to download notes.



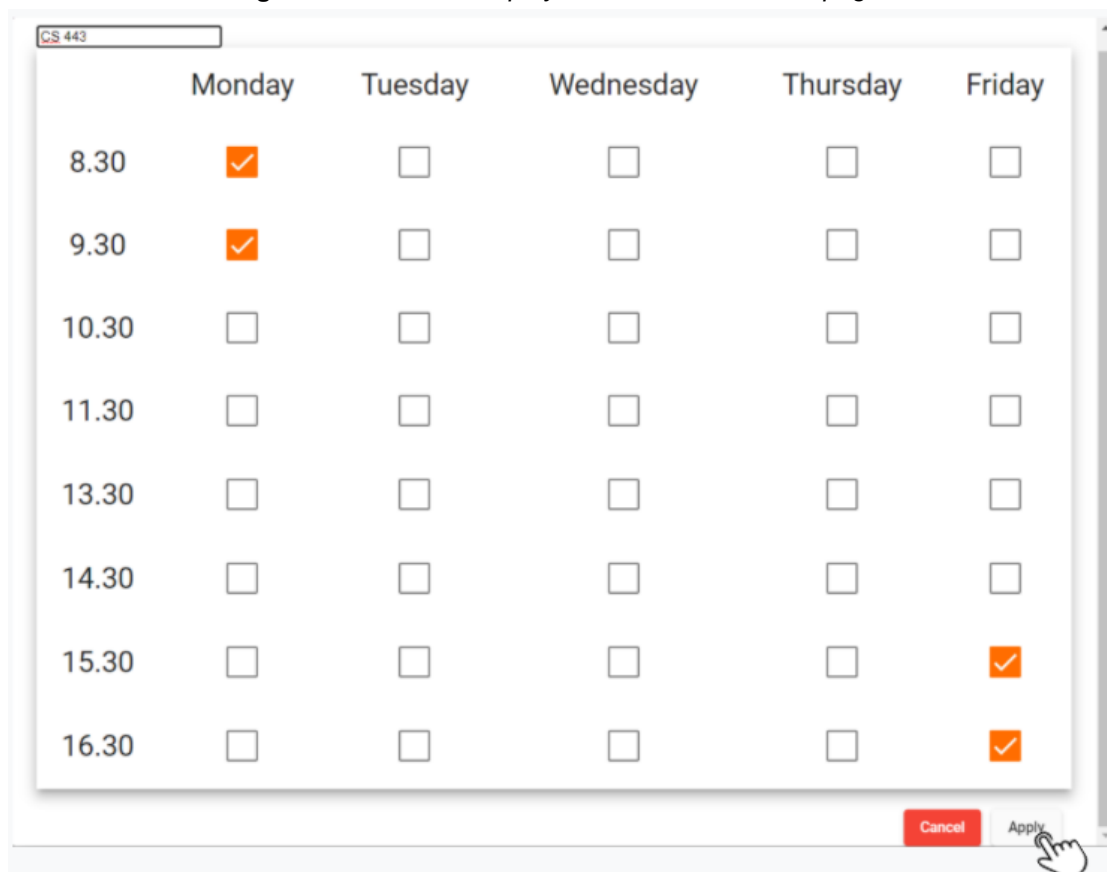
**Figure 12:** Preview and Download buttons for saved notes

### 11.2.3 Add Courses

Instructors can enter a course by clicking on the “+” button displayed above. Once clicked, another screen will open where instructors can enter a course name and select available spots from a calendar by clicking on the time slots. After entering a name and clicking on time slots, instructors can click on the “Apply” button to create that course.



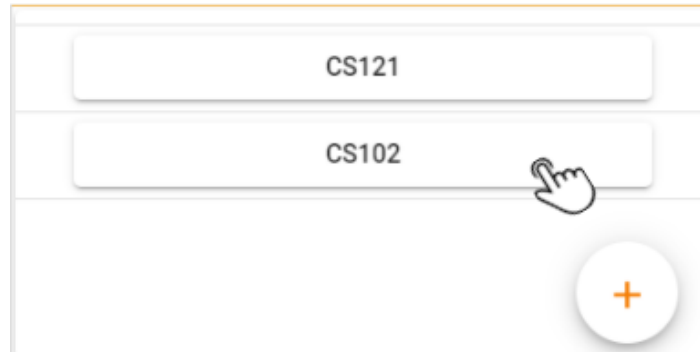
**Figure 13:** Courses displayed in instructor's main page



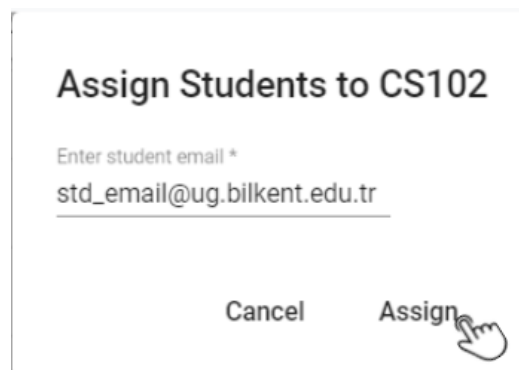
**Figure 14:** Adding new course with selected time slots

### 11.2.4 Assign Student to Courses

Instructors can assign students to their created courses by entering the email of the student and clicking on the “Assign” button. They can open the “Assign Students” section by clicking on the courses that are displayed in the main page.

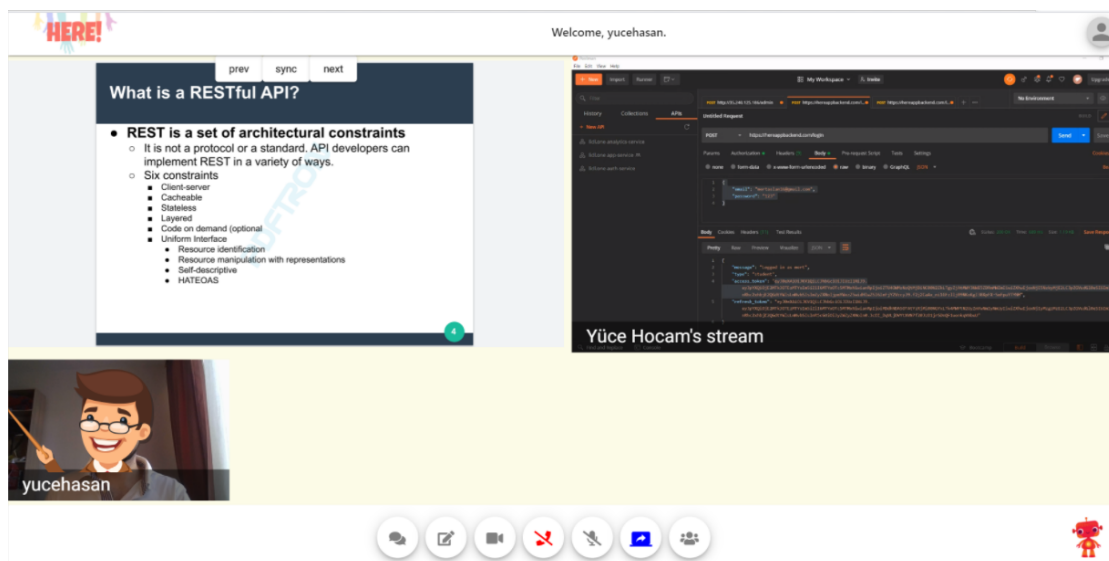


**Figure 15:** Buttons for assigning students to corresponding courses



**Figure 16:** Dialog for assigning students

## 11.3 Conference Page



**Figure 17:** Conference page

### 11.3.1 Conference Utilities

Once the user enters a conference page, they will see a toolbar that displays different utilities for the conference page. From left to right they can click on the button to perform the following actions:

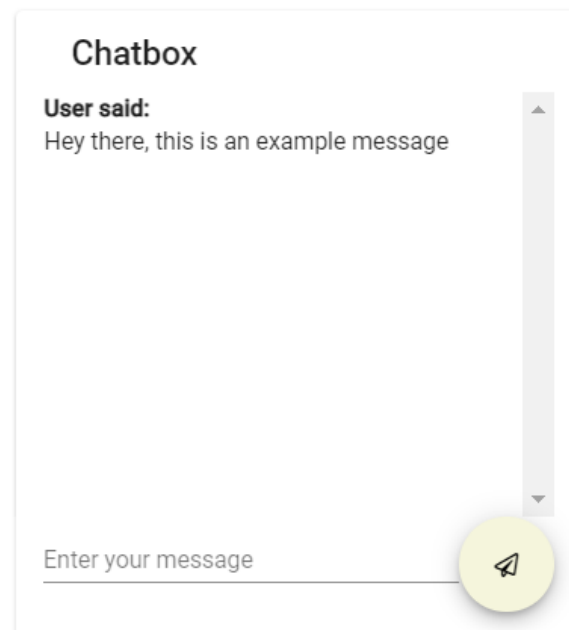
- Open and close the chat panel.
- Open note-taking panel.
- Open or close the camera.
- Leave session.
- Open or close the microphone.
- Start or stop screen sharing
- Show participant list
- Open slide sharing



**Figure 18:** Conference utilities toolbar

### 11.3.2 Chat Panel

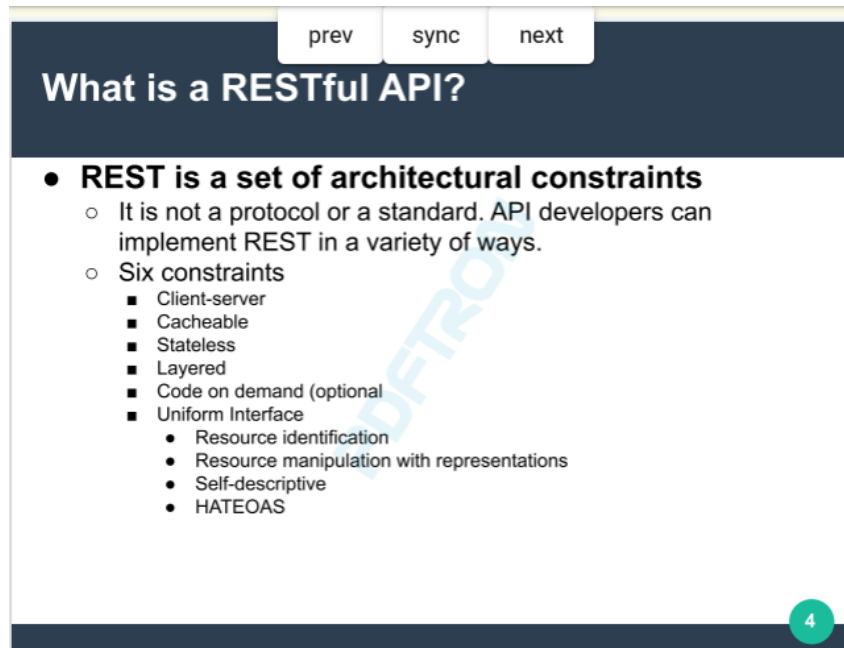
Users can broadcast text messages using the chat panel. There is an input box and a send button next to it.



**Figure 19:** Chat panel

### 11.3.3 Shared Slide

If an instructor starts slide sharing, the slides will be visible for all students on the left panel. Students can click on “prev” or “next” buttons to navigate through the slides, and if they want to go to the slide that the teacher is currently viewing, they can click on the “sync” button, as shown below.



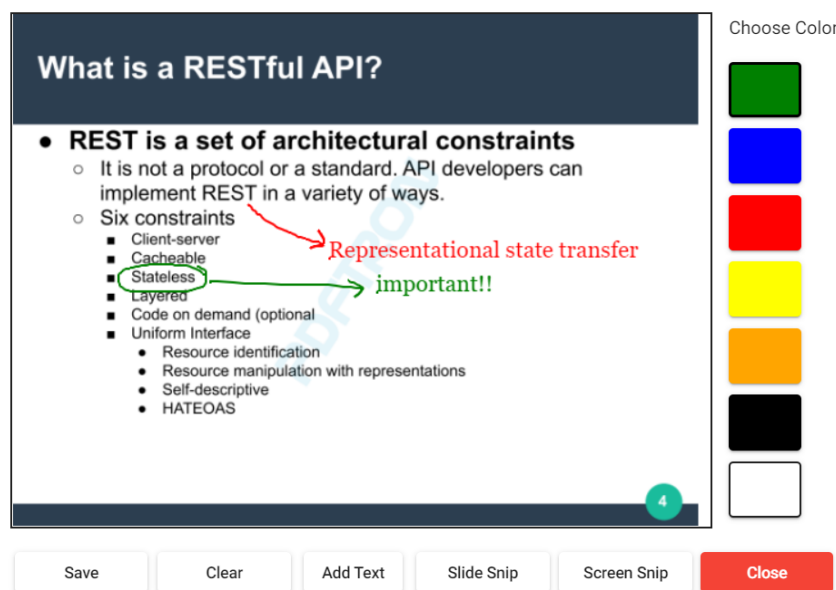
*Figure 20: Shared slide*



*Figure 21: Slide navigation toolbar*

### 11.3.4 Note Taking

Students can open the note-taking tab from the toolbar. Initially, the note-taking panel is an empty canvas. Students can click on the “Slide Snip” button to take the screenshot of the current slide and place it on the canvas. “Screen Snip” button does the same thing but it takes the screenshot of the shared screen panel. Users can then click on the “Add Text” button to add a text panel on the slides where they can type from their keyboards. They can also choose a color on the right and draw something on the canvas as well. “Clear” button clears the canvas and the “Save” button saves the current canvas to the database. To close the canvas, users can click on “Close” button.

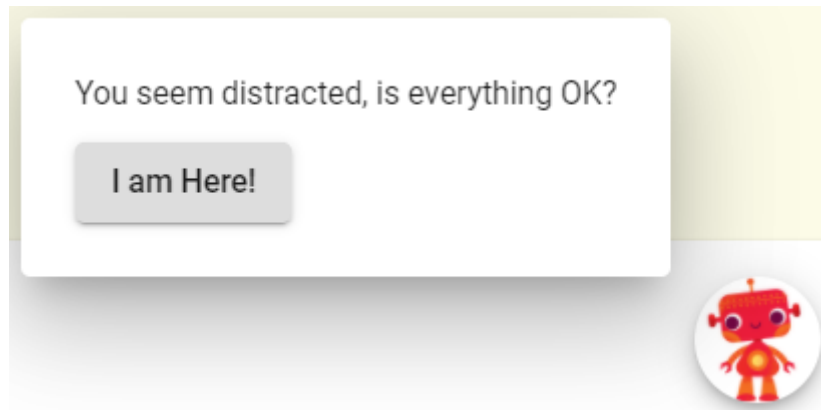


**Figure 22:** Note-taking panel

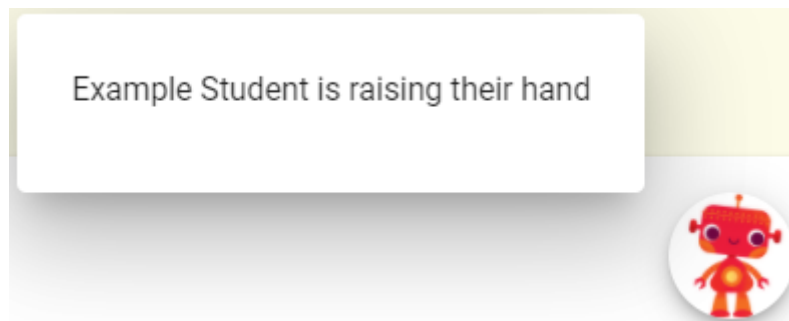


### 11.3.5 TA

Users cannot interact with the TA panel, but the TA panel is important for conveying information to both students and the instructor. For example, if students raised their hands for a question, the instructor will receive feedback from the TA on which students raised their hands. A similar situation occurs for students when they are distracted or they are contributing. When the students receive a notification about them being distracted, they need to confirm that they are listening to the lecture by clicking the “I am here” button.



**Figure 23:** Notification for a student from TA



**Figure 24:** Notification for an instructor from TA

### 11.4 Analytics Page

After the session ends, the analytics page will welcome the instructor. Instructors can see the session details including the number of students distracted between time ranges. They can also see the number of participants, names of the participants and their participation as “normal”, “active” and “inactive” based on the number of times students raise their hands. Instructors can also save this page as a pdf file by clicking on the save button.

Your session has ended. Let's see how it went.

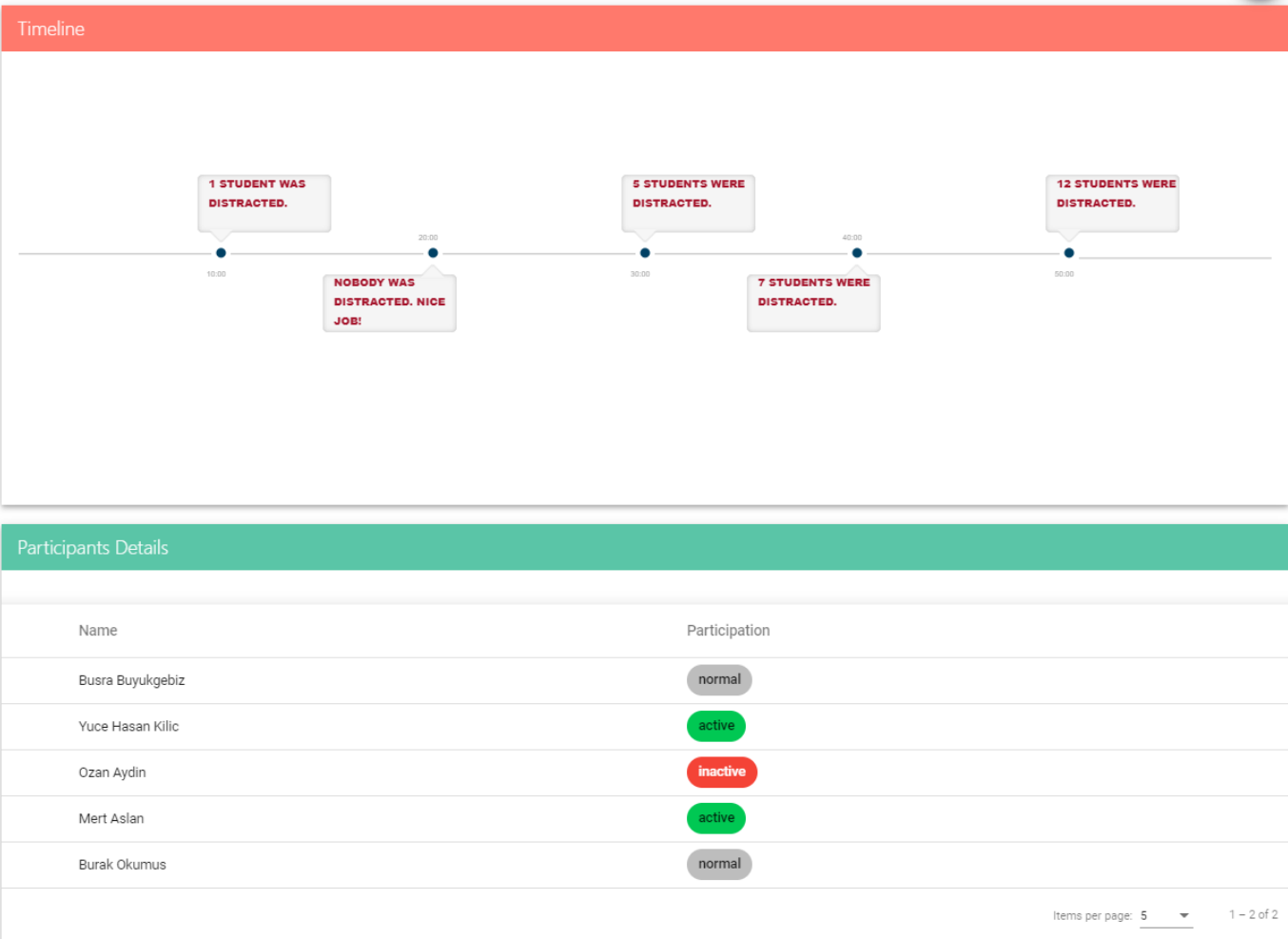


Figure 25: Post-lecture Analytics Screen