Bilkent University
Department of Computer Engineering

Senior Design Project

Project short-name: Here!

Low-level Design Report

Mert Aslan, Rahmiye Büşra Büyükgebiz, Hakkı Burak Okumuş, Ozan Aydın, Yüce Hasan Kılıç

Supervisor: Abdullah Ercüment Çiçek

Jury Members: Çiğdem Gündüz Demir and Can Alkan

Low-level Design Report
Feb 8, 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

Low-level Design 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, we intend to provide a low-level design of the system. First, the object design trade-offs of our system, interface documentation guidelines and engineering standards will be discussed. Then, packages will be examined. Afterwards, we will discuss class interfaces of our project.

1.1 Object design trade-offs

1.1.1 Cost vs. reliability

We decided to use reliable servers such as AWS EC2 for data transfers to prevent loss of data during transfers, crashing of services and to minimize downtime of servers we use. Compared to alternatives such as maintaining servers ourselves, AWS appears to be more expensive but it provides robustness and reliability in return.

1.1.2 Rapid development vs. robustness

Project Here! aims to offer many useful tools in a relatively short period of time. It will not undergo a detailed testing process or will not have early releases. Most of the time will be spent on developing to gain speed and to be able to finish development. As a result, the program may have incapabilities in practice and may require further optimizations along with bug fixes.

1.1.3 Functionality vs. portability

To be able to provide promised functionality, target devices are restricted. Adaptation to other platforms than desktop with different screen resolutions is currently unavailable and this hinders portability.

1.2 Interface documentation guidelines

A sample class is described as follows:

Class Name	Description of the class	
Attributes	attributeName: type	
Methods	methodName(args): return type	Explanation of methods

“Class Name” is replaced with the name of the class, attributes are listed with their names followed by their types and lastly, methods are listed with their signature with a brief explanation on the next column.

1.3 Engineering standards (e.g., UML and IEEE)

This report follows the UML standards to represent class interfaces. Also, IEEE standards for citations are followed throughout the report for all of the references.

1.4 Definitions, acronyms, and abbreviations

- **HTTP:** Hypertext Transfer Protocol
- **AWS:** Amazon Web Services
- **EC2:** Elastic Computing
- **API**: Application Programming Interface
- **TA**: Teaching Assistant
- **UI**: User Interface
- **DB**: Database
- **UML:** Unified Modeling Language
- **IEEE:** Institute of Electrical and Electronics Engineers

2 Packages

2.1 Client

Client package is responsible for operations and visualizations on the user's machine. It has 2 subsystems: View and Controller. View subsystem provides a graphical user interface and receives inputs from the user as well as presenting outputs. User inputs are sent to respective controllers and responses to those inputs are displayed as well. Controller subsystem collects user inputs from View subsystem and invokes requested services such as database manager, server operations etc.

2.1.1 View

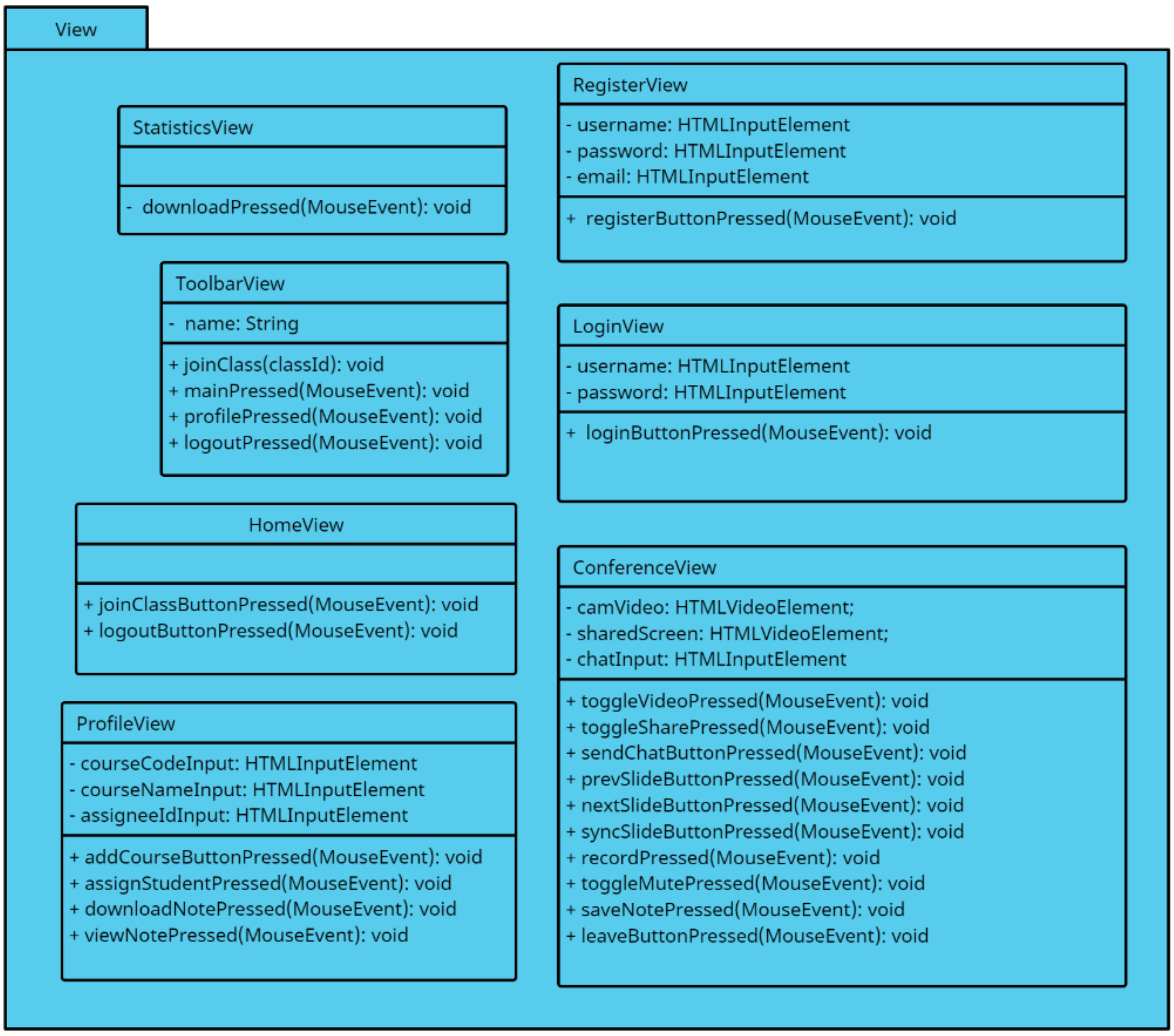

View
StatisticsView
- downloadPressed(MouseEvent): void
ToolbarView
- name: String
+ joinClass(classId): void
+ mainPressed(MouseEvent): void
+ profilePressed(MouseEvent): void
+ logoutPressed(MouseEvent): void
HomeView
+ joinClassButtonPressed(MouseEvent): void
+ logoutButtonPressed(MouseEvent): void
ProfileView
- courseCodeInput: HTMLInputElement
- courseNameInput: HTMLInputElement
- assigneeIdInput: HTMLInputElement
+ addCourseButtonPressed(MouseEvent): void
+ assignStudentPressed(MouseEvent): void
+ downloadNotePressed(MouseEvent): void
+ viewNotePressed(MouseEvent): void
RegisterView
- username: HTMLInputElement
- password: HTMLInputElement
- email: HTMLInputElement
+ registerButtonPressed(MouseEvent): void
LoginView
- username: HTMLInputElement
- password: HTMLInputElement
+ loginButtonPressed(MouseEvent): void
ConferenceView
- camVideo: HTMLVideoElement;
- sharedScreen: HTMLVideoElement;
- chatInput: HTMLInputElement
+ toggleVideoPressed(MouseEvent): void
+ toggleSharePressed(MouseEvent): void
+ sendChatButtonPressed(MouseEvent): void
+ prevSlideButtonPressed(MouseEvent): void
+ nextSlideButtonPressed(MouseEvent): void
+ syncSlideButtonPressed(MouseEvent): void
+ recordPressed(MouseEvent): void
+ toggleMutePressed(MouseEvent): void
+ saveNotePressed(MouseEvent): void
+ leaveButtonPressed(MouseEvent): void

Figure x: Class Diagram of View Subsystem

HomeView: Used to display the home screen after the user has successfully logged in.

ProfileView: Used to display the profile details for the current users. Other than seeing the profile details, students can also see their saved notes on this page and instructors can see their courses and add students to them.

ConferenceView: Used to display the shared screen, shared slides, video of hosts and video of participants during a lecture.

LoginView: Used to display a screen that enables existing users login.

RegisterView: Used to display a screen that enables new users to register.

StatisticsView: Used to display the post-lecture statistics to the instructor.

ToolbarView: Used to display the navigation buttons for the main page, profile page and logout.

2.1.2 Controller

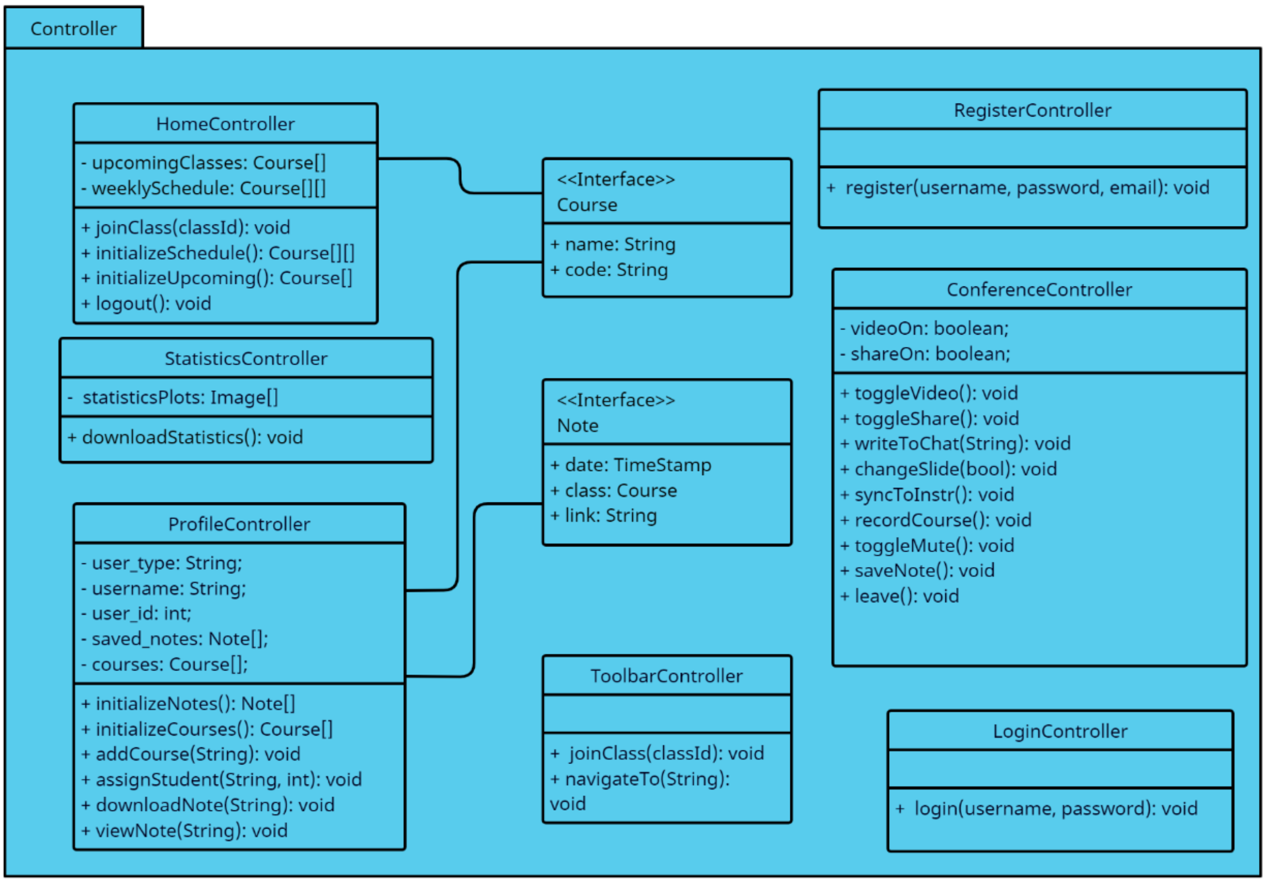


Figure x: Class Diagram of Controller Subsystem

<interface> Course: Used to store the code and the name information of the courses.

<interface> Note: Used to store the date, class and the link information of the notes.

HomeController: Used to retrieve upcoming classes and weekly schedule information for the current user from the database, send it to the view. Join class and logout functions are also managed by HomeController.

ProfileController: Used to retrieve data about the current user from the database and send it to view, as well as adding or updating such data if needed.

ConferenceController: Used to manage the current conference according to the user input.

LoginController: Used to send the login info to the database and retrieve a login response.

RegisterController: Used to send the register info to the database and retrieve a register response.

StatisticsController: Used to generate downloadable post-lecture statistics.

ToolbarController: Used to navigate to the relevant page when a button on the toolbar is pressed.

2.2 Server

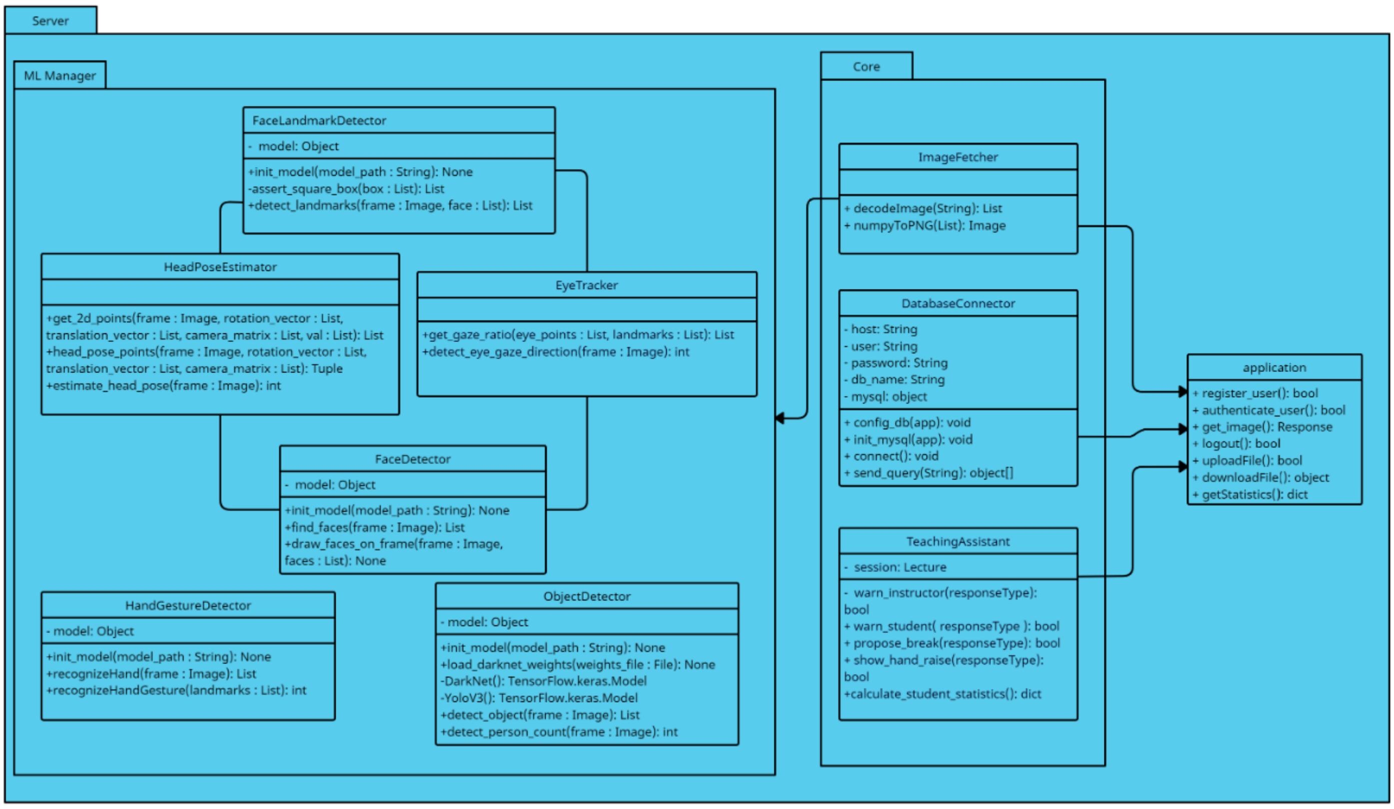

Server
ML Manager
FaceLandmarkDetector
- model: Object
+init_model(model_path : String): None
-assert_square_box(box : List): List
+detect_landmarks(frame : Image, face : List): List
HeadPoseEstimator
+get_2d_points(frame : Image, rotation_vector : List, translation_vector : List, camera_matrix : List, val : List): List
+head_pose_points(frame : Image, rotation_vector : List, translation_vector : List, camera_matrix : List): Tuple
+estimate_head_pose(frame : Image): int
EyeTracker
+get_gaze_ratio(eye_points : List, landmarks : List): List
+detect_eye_gaze_direction(frame : Image): int
FaceDetector
- model: Object
+init_model(model_path : String): None
+find_faces(frame : Image): List
+draw_faces_on_frame(frame : Image, faces : List): None
HandGestureDetector
- model: Object
+init_model(model_path : String): None
+recognizeHand(frame : Image): List
+recognizeHandGesture(landmarks : List): int
ObjectDetector
- model: Object
+init_model(model_path : String): None
+load_darknet_weights(weights_file : File): None
-DarkNet(): TensorFlow.keras.Model
-YoloV3(): TensorFlow.keras.Model
+detect_object(frame : Image): List
+detect_person_count(frame : Image): int
Core
ImageFetcher
+ decodeImage(String): List
+ numpyToPNG(List): Image
DatabaseConnector
- host: String
- user: String
- password: String
- db_name: String
- mysql: object
+ config_db(app): void
+ init_mysql(app): void
+ connect(): void
+ send_query(String): object[]
TeachingAssistant
- session: Lecture
- warn_instructor(responseType): bool
+ warn_student(responseType): bool
+ propose_break(responseType): bool
+ show_hand_raise(responseType): bool
+calculate_student_statistics(): dict
application
+ register_user(): bool
+ authenticate_user(): bool
+ get_image(): Response
+ logout(): bool
+ uploadFile(): bool
+ downloadFile(): object
+ getStatistics(): dict

Figure x: Class Diagram of Server Subsystem

2.2.1 ML Manager

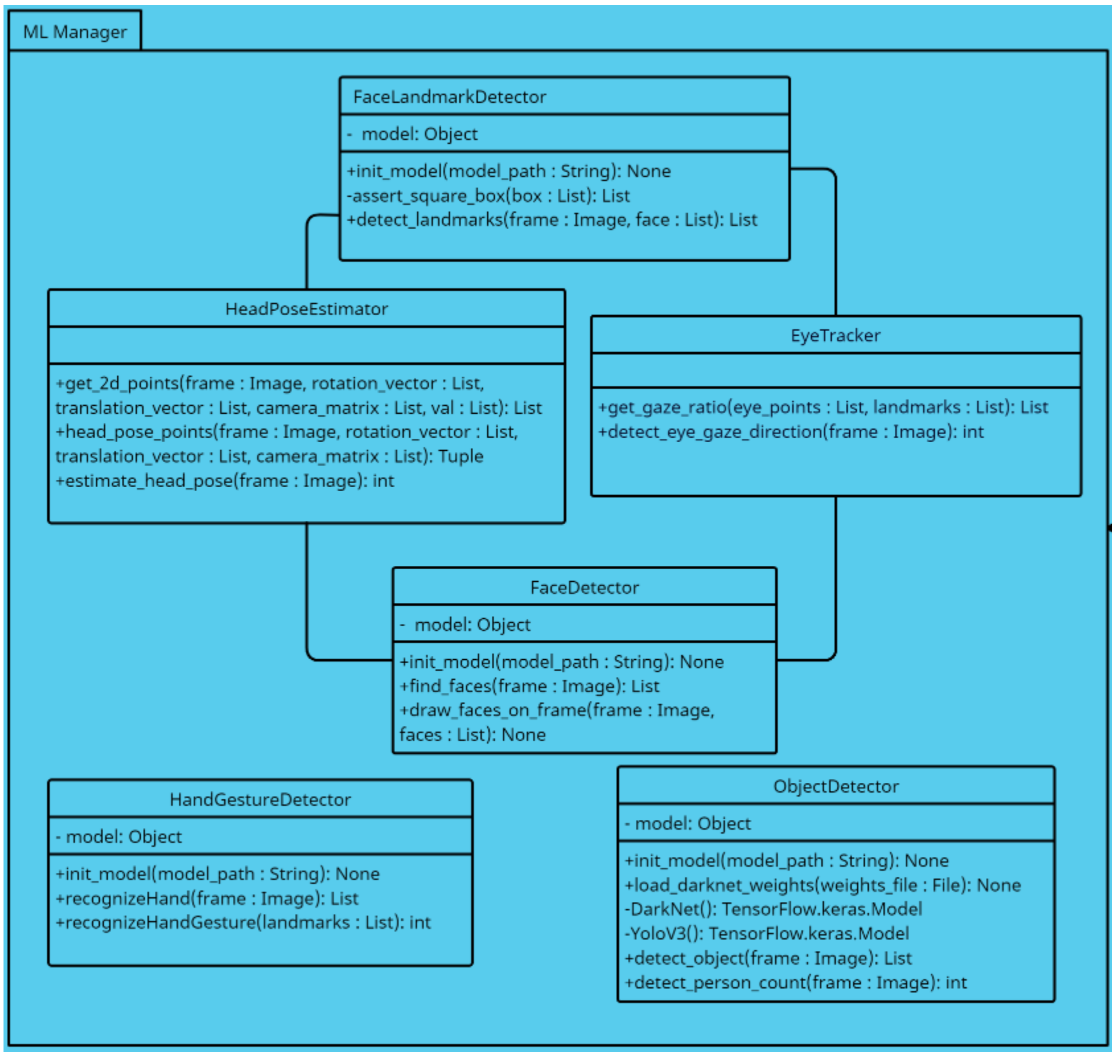

ML Manager
FaceLandmarkDetector
- model: Object
+init_model(model_path : String): None
-assert_square_box(box : List): List
+detect_landmarks(frame : Image, face : List): List
HeadPoseEstimator
+get_2d_points(frame : Image, rotation_vector : List, translation_vector : List, camera_matrix : List, val : List): List
+head_pose_points(frame : Image, rotation_vector : List, translation_vector : List, camera_matrix : List): Tuple
+estimate_head_pose(frame : Image): int
EyeTracker
+get_gaze_ratio(eye_points : List, landmarks : List): List
+detect_eye_gaze_direction(frame : Image): int
FaceDetector
- model: Object
+init_model(model_path : String): None
+find_faces(frame : Image): List
+draw_faces_on_frame(frame : Image, faces : List): None
HandGestureDetector
- model: Object
+init_model(model_path : String): None
+recognizeHand(frame : Image): List
+recognizeHandGesture(landmarks : List): int
ObjectDetector
- model: Object
+init_model(model_path : String): None
+load_darknet_weights(weights_file : File): None
-DarkNet(): TensorFlow.keras.Model
-YoloV3(): TensorFlow.keras.Model
+detect_object(frame : Image): List
+detect_person_count(frame : Image): int

Figure x: Class Diagram of ML Manager Subsystem

FaceDetector: Detects faces on a given frame.

FaceLandmarkDetector: Detects 68 facial landmarks on a given frame.

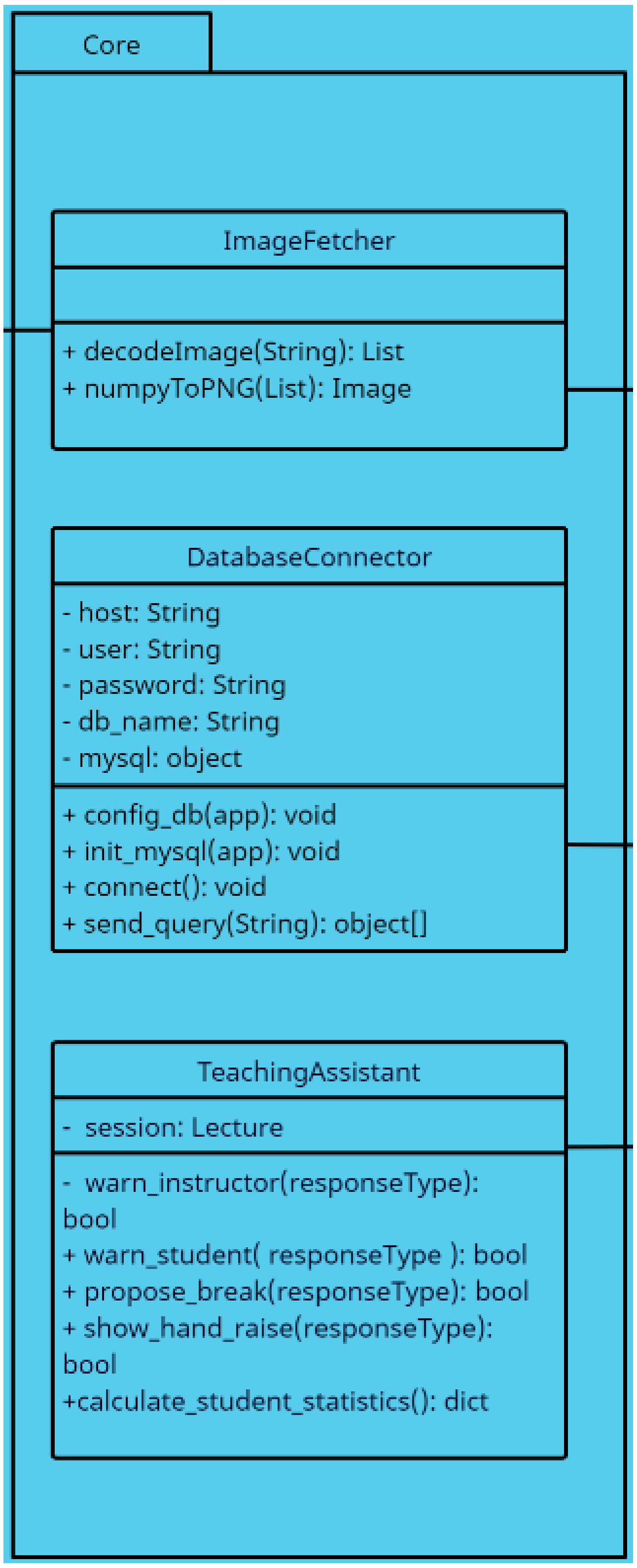
HeadPoseEstimator: Estimates the direction of a face in a given frame.

EyeTracker: Estimates the direction a person is looking at in a given frame.

HandGestureDetector: Detects hands in a given frame and gestures made by them.

ObjectDetector: Detects the presence of an object and a person in a given frame.

2.2.2 Core



Core
ImageFetcher
+ decodeImage(String): List
+ numpyToPNG(List): Image
DatabaseConnector
- host: String
- user: String
- password: String
- db_name: String
- mysql: object
+ config_db(app): void
+ init_mysql(app): void
+ connect(): void
+ send_query(String): object[]
TeachingAssistant
- session: Lecture
- warn_instructor(responseType): bool
+ warn_student(responseType): bool
+ propose_break(responseType): bool
+ show_hand_raise(responseType): bool
+calculate_student_statistics(): dict

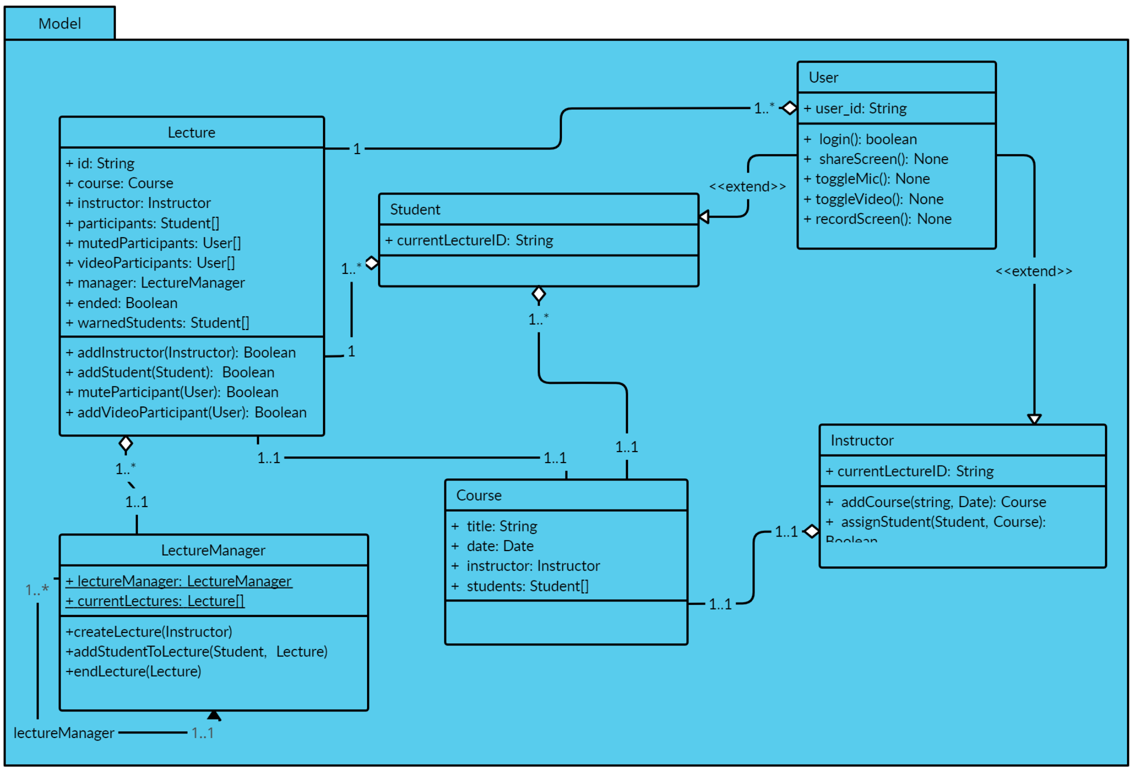
Figure x: Class Diagram of Core Subsystem

ImageFetcher: Responsible for transferring and modifying the image data from frontend to backend.

DatabaseConnector: Responsible for connecting to the database, as well as sending queries and receiving the return objects.

TeachingAssistant: Responsible for calculating statistics for students, interacting with the students or instructors using these information.

2.2.3 Model



Model
User
+ user_id: String
+ login(): boolean
+ shareScreen(): None
+ toggleMic(): None
+ toggleVideo(): None
+ recordScreen(): None
Lecture
+ id: String
+ course: Course
+ instructor: Instructor
+ participants: Student[]
+ mutedParticipants: User[]
+ videoParticipants: User[]
+ manager: LectureManager
+ ended: Boolean
+ warnedStudents: Student[]
+ addInstructor(Instructor): Boolean
+ addStudent(Student): Boolean
+ muteParticipant(User): Boolean
+ addVideoParticipant(User): Boolean
Student
+ currentLectureID: String
<<extend>>
<<extend>>
Instructor
+ currentLectureID: String
+ addCourse(string, Date): Course
+ assignStudent(Student, Course): Boolean
Course
+ title: String
+ date: Date
+ instructor: Instructor
+ students: Student[]
LectureManager
+ lectureManager: LectureManager
+ currentLectures: Lecture[]
+createLecture(Instructor)
+addStudentToLecture(Student, Lecture)
+endLecture(Lecture)
lectureManager
1
1..*
1..*
1
1..*
1..*
1..1
1..1
1..1
1..1
1..1
1..1
1..1
1..*
1..1

Figure x: Class Diagram of Model Subsystem

Lecture: Lecture class that contains the necessary information for a single session.

Course: Course class that contains the available course information like students taking that course.

Student: Student class that will hold the student information for easy access, instead of querying the database each time.

LectureManager: Singleton class, holds instances of the current active classes and manages them. Its functionality will be expanded as the development continues.

User: Representative class for the users.

Instructor: Instructor class that will hold the instructor information for easy access, instead of querying the database each time.

3 Class Interfaces

3.1 Client

3.1.1 View

HomeView	Displays the home screen after the user has successfully logged in.	
Attributes	None	
Methods	joinClassButtonPressed(MouseEvent): void logoutButtonPressed(MouseEvent) :void	Invokes handler of "Join Class" button Invokes handler of "Logout" button

ProfileView	Displays the profile details for the current user. Other than seeing the profile details, students can also see their saved notes on this page and instructors can see their courses and add students to them.
Attributes	courseCodeInput: HTMLInputElement courseNameInput: HTMLInputElement assigneeIdInput: HTMLInputElement	
Methods	addCourseButtonPressed(MouseEvent): void	Sends course code and course name inputs to the controller to add a new course with that code and name
	assignStudentPressed(MouseEvent): void	Sends course code and assignee id inputs to the controller to assign student with corresponding id to the course with given code
	downloadNotePressed(MouseEvent): void	Invokes handler of "Download Note" button
	viewNotePressed(MouseEvent): void	Opens o popup to view the saved note

ConferenceView	Displays the shared screen, shared slides, video of hosts and video of participants during a lecture.	
Attributes	camVideo: HTMLVideoElement sharedScreen: HTMLVideoElement chatInput: HTMLInputElement	
Methods	toggleVideoPressed(MouseEvent): void	Invokes handler of "Toggle Video" button and updates camVideo element
	toggleSharePressed(MouseEvent): void	Invokes handler of "Toggle Share" button and updates sharedScreen element
	sendChatButtonPressed(MouseEvent) : void	Sends chat input to the controller to send a message
	prevSlideButtonPressed(MouseEvent) : void	Invokes handler of "Previous Slide"

	nextSlideButtonPressed(MouseEvent) : void syncSlideButtonPressed(MouseEvent) : void recordPressed(MouseEvent): void toggleMutePressed(MouseEvent): void saveNotePressed(MouseEvent): void leaveButtonPressed(MouseEvent): void	button Invokes handler of “Next Slide” button Invokes handler of “Synchronize with Instructor” button Invokes handler of “Record Lesson” button Invokes handler of “Toggle Mic” button Converts the note taken and sends it to controller Invokes handler of “Leave” button

LoginView	Displays a screen that enables existing users login.	
Attributes	username: HTMLInputElement password: HTMLInputElement	
Methods	loginButtonPressed(MouseEvent): void	Sends username and password inputs to the controller to log in.

RegisterView	Displays a screen that enables new users to register.	
Attributes	username: HTMLInputElement password: HTMLInputElement email: HTMLInputElement	
Methods	registerButtonPressed(MouseEvent): void	Sends email, username and password inputs to the controller to register a new account

StatisticsView	Displays the post-lecture statistics to the instructor.	
Attributes	None	
Methods	downloadPressed(MouseEvent): void	Invokes handler of “Download Statistics” button

ToolbarView	Displays the navigation buttons for the main page, profile page and logout.	
Attributes	name: String	
Methods	mainPressed(MouseEvent): void profilePressed(MouseEvent): void	Invokes handler of “Main” button Invokes handler of “Profile” button

	logoutPressed(MouseEvent): void	Invokes handler of “Logout” button

3.1.2 Controller

<<Interface>> Course	An interface for representing course data.
Attributes	name: String code: String

<<Interface>> Note	An interface for representing a taken note data.
Attributes	date: Timestamp class: Course link: String

HomeController	Handles operations required by HomeView such as retrieving upcoming classes and weekly schedule information for the current user from the database. Join class and logout functions are also managed by HomeController.	
Attributes	upcomingClasses: Course[] weeklySchedule: Course[][]	
Methods	joinClass(classID): void	?
	initializeSchedule(): Course[][]	Fetches weekly schedule of user from database
	initializeUpcoming(): Course[]	Fetches upcoming classes of user from database
	logout(): void	Logs user out

ProfileController	Handles operations required by ProfileView such as retrieving data about the current user from the database as well as adding or updating such data if needed.	
Attributes	user_type: String username: String user_id: int saved_notes: Note[] courses: Course[]	
Methods	initializeNotes(): Note[]	Fetches saved notes from database if the user is a student
	initializeCourses(): Course[]	Fetches registered courses from database if the user is an instructor
	addCourse(String): void	Registers a new course with a given name and code if user is instructor

	assignStudent(String, int): void downloadNote(String): void viewNote(String): void	Assigns the student with given id to the course with given course code. Downloads a saved note from the given link Previews a saved note from the given link

Conference Controller	Provides in-conference controls for user media input, shared slide and note-taking feature.	
Attributes	videoOn: boolean shareOn: boolean	
Methods	toggleVideo(): void	Gets user’s webcam video media
	toggleShare(): void	Gets user’s screen media
	writeToChat(String): void	Send given message to the chat server
	changeSlide(boolean): void	Skips shared slide forward or backwards in the client
	syncToInstr(): void	Synchronizes shared slide with the slide the instructor is currently viewing
	recordCourse(): void	Starts recording current lesson
	toggleMute(): void	Toggles user’s mic input
	saveNote(): void	Stores the saved note into the database
	leave(): void	Disconnects from the current lesson

LoginController	Establishes connection with the database to provide authentication for the user.	
Attributes	None	
Methods	login(username: String, password: String): void	Sends login request to the database server

Register Controller	Establishes connection with the database to create a new account.	
Attributes	None	
Methods	register(username: String, password: String, email: String): void	Sends register request to the database server

Statistics Controller	Handles downloading of the statistics file after a lesson.

Attributes	statisticsPlots: Image[]	
Methods	downloadStatistics(): void	Downloads statistics to the computer

ToolbarController	Handles navigation among the pages of the application.	
Attributes	None	
Methods	navigateTo(String): void	Navigates to the page with the given path

3.2 Server

3.2.1 Application

Application	This is the main class of the server which accepts the incoming HTTP requests and returns corresponding responses. This class utilizes Flask framework and has several endpoints for different requests. Parameters of incoming HTTP requests are passed through FormData API by Angular. This class is structured as a router class that calls core methods for processing data and passing/retrieving data to/from database.	
Attributes	None	
Methods	register_user(): bool	Registers a user with the given credential info.
	authenticate_user(): bool	Checks the given user credentials for login.
	get_image(): Response	Returns a response to an image data, image data is passed to the Core package and processed by ML Manager.
	logout(): bool	Logs out user.
	uploadFile(): bool	Uploads a file to the servers directory.
	downloadFile(): object	Returns a file from the servers directory.
	getStatistics():	Returns statistics of the Lecture when lecture ends.

Methods of the Application class will be expanded as the program development goes on. There will be more HTTP requests that will need to be responded.

3.2.2 ML Manager

FaceDetector	Responsible for detecting faces in a given frame. These faces are later on used in other classes (HeadPoseEstimator and EyeTracker) in order to track the attention of the students.
Attributes	model: Object

Methods	init_model(model_path: String): None	Initializes the face detector model.
	find_faces(frame: Image) List	Finds faces on a given frame using the model stored in the class.
	draw_faces_on_frame(frame: Image, faces: List): None	Draws the faces using OpenCV to the screen. (for testing purposes)

FaceLandmark Detector	Responsible for detecting facial landmarks in a given frame. These landmarks are later on used in other classes (HeadPoseEstimator and EyeTracker) in order to track the attention of the students.	
Attributes	model: Object	
Methods	init_model(model_path: String): None	Initializes the facial landmark detector model.
	assert_square_box(box : List): List	Checks if the box obtained by the face detector is a square or not. Enlarges the box into a square if it is not.
	detect_landmarks(frame: Image, face: List): List	Detects 68 facial landmarks on a face that is in a given frame.

HeadPoseEstimator	Estimates the direction the head in a given frame is looking towards. This is used in order to be able to track the attention of the student to the lecture. This class makes use of FaceDetector and FaceLandmarkDetector classes in order to function and estimate the head pose.	
Attributes	None	
Methods	head_pose_points(frame: Image, rotation_vector: List, translation_vector: List, camera_matrix: List): Tuple	Gets the points on the frame to estimate a head that is facing sideways.
	get_2d_points(frame: Image, rotation_vector: List, translation_vector: List, camera_matrix: List, val: List): List	Projects the 3D points onto 2D to be able to produce
	estimate_head_pose(frame: Image): int	Estimates the direction of the head in a given frame is looking towards by making use of the functions mentioned above.

EyeTracker	Detects the gaze direction of a student using a given frame. This class makes use of the FaceDetector and FaceLandmarkDetector classes.	
Attributes	None	
Methods	get_gaze_ratio(eye_points: List, landmarks: List): List	Divides the eye in a given frame into two. Then checks the ratio of the white

		region of the eye (the sclera) to the colored part (the iris) on both of these parts. Deduces that a person is looking towards a certain direction if the difference in these ratios are above a certain threshold.
	detect_eye_gaze_direction(frame: Image)	Detects whether a student is paying attention to the lecture or not by estimating their gaze direction.

HandGesture Detector	Detects hands, fingers and gestures made by these hands in a given frame. This class is used to be able to understand if a student wants to ask a question by raising their hand.	
Attributes	model: Object	
Methods	init_model(model_path: String): None	Initializes the hand and finger detector model.
	recognizeHand(frame : Image): List	Detects hands and fingers on a given frame using the model of the class.
	recognizeHandGesture(landmarks : List): int	Detects the hand gesture made by the hand in the given frame using the detected hand landmarks by the recognizeHand function.

ObjectDetector	Detects objects (phones in particular) in a given frame. Also detects if a person is visible in front of the camera to be able to detect student presence.	
Attributes	model: Object	
Methods	init_model(model_path : String): None	Initializes the DarkNet and YoloV3 convolutional neural networks.
	load_darknet_weights(weights_file : File): None	Loads the necessary DarkNet weights.
	DarkNet(): TensorFlow.keras.Model	Initializes the DarkNet CNN.
	YoloV3(): TensorFlow.keras.Model	Initializes the YoloV3 CNN.
	detect_object(frame : Image): List	Detects objects in a given frame.
	detect_person_count(frame : Image): int	Counts the number of people that are present in a given frame.

3.2.3 Core

ImageFetcher	Responsible for transferring and modifying the image data from frontend to backend.

Attributes	None	
Methods	decodeImage(String): List numpyToPNG(List): Image	Takes an image in string format and returns a Numpy array. Takes a numpy array and turns it into a PNG image.

DatabaseConnector	Responsible for connecting to the database, as well as sending queries and receiving the return objects.	
Attributes	host: String user: String password: String db_name: String mysql: Object	
Methods	config_db(app): void init_mysql(app): void connect(): void send_query(String): Object[]	Load the db config info to the server. Initialize the mysql server using the information from the config file. Connect to the initialized mysql server. Send a query to the server, return the resulting object as a list.

TeachingAssistant	Responsible for calculating statistics for students, interacting with the students or instructors using these information.	
Attributes	session: Lecture	
Methods	warn_instructor(responseType): bool warn_student(responseType): bool propose_break(responseType): bool show_hand_raise(responseType):bool calculate_student_statistics(): dict	Prompts the instructor for various reasons. Prompts the instructor for various reasons. Propose a break if too much distraction is detected. Show the students who have their hands raised to the instructor. Calculate the statistics of the students in the lecture.

3.2.4 Model

Lecture	Lecture class that contains the necessary information for a single session.	
Attributes	id: String course: Course instructor: Instructor participants: Student[] mutedParticipants: User[] videoParticipants: User[] manager: LectureManager ended: bool warnedStudents: Student[]	
Methods	addInstructor(Instructor): bool addStudent(Student): bool	Adds an instructor to the lecture. Adds an instructor to the lecture.

	muteParticipant(User): bool addVideoParticipant(User): bool	Mutes a participant.

LectureManager	Singleton class, holds instances of the current active classes and manages them. Its functionality will be expanded as the development continues.	
Attributes	lectureManager: LectureManager currentLectures: Lecture[]	
Methods	createLecture(Instructor): Lecture addStudentToLecture(Student, Lecture): bool endLecture(Lecture): bool	Creates a lecture. Adds a student to the lecture. Ends a lecture.

Student	Holds the student information for easy access, instead of querying the database each time.
Attributes	currentLectureID: String
Methods	None

Course	Course class that contains the available course information like students taking that course.
Attributes	title: String date: Date instructor: Instructor students: Student[]
Methods	None

User	Representative class for the users.	
Attributes	user_id: String	
Methods	login(): bool shareScreen(): void toggleMic(): void toggleVideo(): void	Checks if the user is logged in. Allows users to share his screen. Toggles mic. Toggles video.

Instructor	Holds the instructor information for easy access, instead of querying the database each time.
Attributes	currentLectureID: String

Methods	addCourse(String, Date): Course assignStudents(Student, Course): bool	Instructor can add a course to the system. Instructor can assign students to the courses.

4 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 [3].
- **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 [4].

5 References

[1] "Cloud computing with AWS". [Online]. Available: https://aws.amazon.com/what-is-aws/. [Accessed: 07-Feb-2021].

[2] "Amazon EC2". [Online]. Available: https://aws.amazon.com/ec2/. [Accessed: 07-Feb-2021].

[3] "Welcome to Flask¶," *Welcome to Flask - Flask Documentation (1.1.x)*. [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/. [Accessed: 07-Feb-2021].

[4] "Angular". [Online]. Available: https://angular.io/. [Accessed: 07-Feb-2021].