Bilkent University

Department of Computer Engineering

CS 492 - Senior Design Project II

Project Name: PANDETECT

Low Level Design Report

Group Members: Selen Uysal, Berk Güler, İrem Seven, Ufuk Bombar, Batuhan Tosyalı

Supervisor: Selim Aksoy

Innovation Expert: Ahmet Eren Başak

1. Introduction

Due to the COVID-19 pandemic which we have faced in 2020, there have been major changes in our lifestyle. Since COVID-19 is a virus that can spread very easily, it affected many people all around the world. In addition to the easy spread of the virus, it poses a big problem due to its high lethal effect. There are some rules that people must follow from the beginning of the pandemic in order to protect themselves and society from the virus. Two of the most important of these are wearing masks and keeping social distance. For this reason, states require wearing masks and impose curfews at regular intervals. This means that although the rules are tried to be applied, there are cases where the mask and social distance control cannot be fully achieved. Therefore, we aimed to make a system to prevent these situations..

PANDETECT is an integrated system that will detect whether people in a particular area obey social distance rules and whether they wear masks or not. Being an integrated system, PANDETECT consists of two separate applications one in the form of a desktop and one in the form of a mobile application. Our system aims to extract data about whether the rules are followed or not in specific locations in order to present it to users in the mobile application's map view. All users who want to take advantage of our system can choose to stay away from areas that could put their health at risk. Business owners, on the other hand, can use our system if they want to have more effective control in their spaces, or if they want to provide data to their customers as an indication of the safety levels of their places. With the help of the desktop application, they can watch the stream of cameras, while getting instant data, which will enable them easy and effective control. Likewise, individuals or institutions responsible for regulations can use our system to audit the people and places more effectively.

To keep a safe environment during the pandemic, governments have the duty of inspecting the public places and closed areas such as restaurants, to see whether they comply with the COVID-19 regulations to the law enforcement officers but it requires a lot of ground-work and it is known that humans are more error-prone compared to a machine. In order to maximize the efficiency of the regulations, the PANDETECT system will continuously check whether the regulations are being complied with, and statistics will be generated according to the camera device's field of view.

The main goal of the mobile application is to identify places, by extracting data, that do not comply with the pandemic rules such as restaurants, workplaces, and public areas. In this way, it is expected to increase the control in the places where do not comply with the rules. This information will be shared on a map in the PANDETECT mobile application using the data of the places where rules are followed and not followed. That way people can choose not to go to areas where the rules are not followed according to the map provided. Thus, our system aims to inform users about compliance with the rules in some specific places. Also, the provided data can be used to increase regulations by the government. Similarly, business owners can use the system to make sure they have control over their places. With the help of the desktop application, it will be easier for them to control the violation situations. That way it will be easier for them to keep a healthy environment.

It is important to note that although PANDETECT will be developed with COVID-19 in mind, it will be a system that may be used for other pandemics in the coming years. The spread of viruses is very similar regardless of their types. Thus, we aimed to use our system for not only the COVID-19 pandemic but also for future pandemic conditions. Also, it is important for us to implement real-life solutions with computer vision concepts. We believe that implementing such real-life solutions will contribute to intellectual and scientific knowledge.

1.1 Object Design Trade-offs

1.1.1 Usability vs Accuracy

While using the face detection and distance detection algorithms in real-time, we may need to sacrifice some of the processing power which can be called accuracy to increase the usability of the face detection algorithms. With that, our algorithm can respond to the real time requirements.

1.1.2 Flexibility vs Usability

We have decided to use ESP-32 CAM devices (AI thinker Model) in order to build a real-time embedded application in order to increase flexibility of the application. To give an example, with the new devices, we can log to new wi-fi’s with the usage of bluetooth therefore it increased the usability, but we have sacrificed the usability, since the cameras of the ESP-32 CAM devices are only 2 MP which decreases the usability of the system since we need to carefully determine where we need to put the new camera devices.

1.1.3 Performance vs Privacy

Since we included some privacy methods for the business owners such as authentication, we have to sacrifice some of the performance in the backend to increase the privacy of our users, while giving some performance from SQL database.

1.1.4 Compatibility vs Programmability

Since we are working with multiple platforms (camera device, PC, mobile), we need to write proper code in order to sustain the requirements of all of those platforms we included in our system. Therefore, we need to increase compatibility which decreases our proper programmability since we need to equip new technologies to further balance our compatibility such as using TypeScript.

1.2 Interface Documentation Guidelines

In the documentation, all class names are singular and named with standard class name form like ‘ClassName’. Furthermore variable methods names follow the same form like ‘variable’ and ‘method()’. The description of the class starts with the class name followed by the method names and variables

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

We followed the IEEE citation format for referencing our resources and UML design principles for class descriptions and diagrams.

1.4 Definitions, acronyms, and abbreviations

TERM	**Definitions acronyms and abbreviations**
MVC	Model view controller is a pattern that separates the application to three other main logical components, model ,view , controller. The separation allows us to work better and separately not depending on other developers for specific parts.

CPU	Central processing unit
ESP-32 CAM DEVICE	This device is an embedded development board devoted for camera applications
UML	Unified Modeling Language
IEEE	Institute of Electrical and Electronics Engineers
Google Maps API	An api distributed from google that allows us to use the google maps screen in our system’s mobile part
Postgres	An sql for unix that we use.

2. Packages

In this section, firstly, the subsystem decomposition of the device system, backend system and mobile application are given. Secondly, the packages are explained in detail.

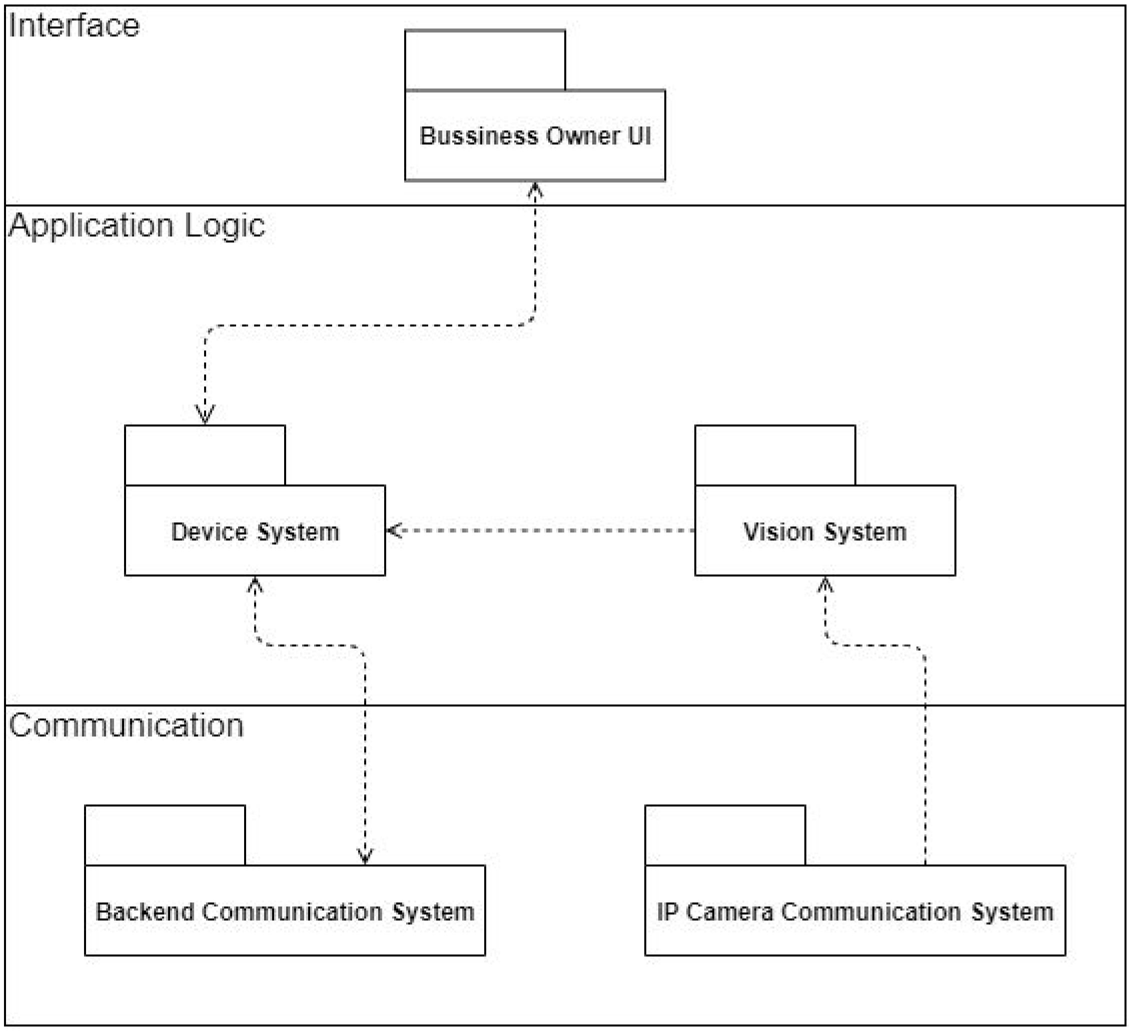

Figure 1: Subsystem Decomposition of the Device System

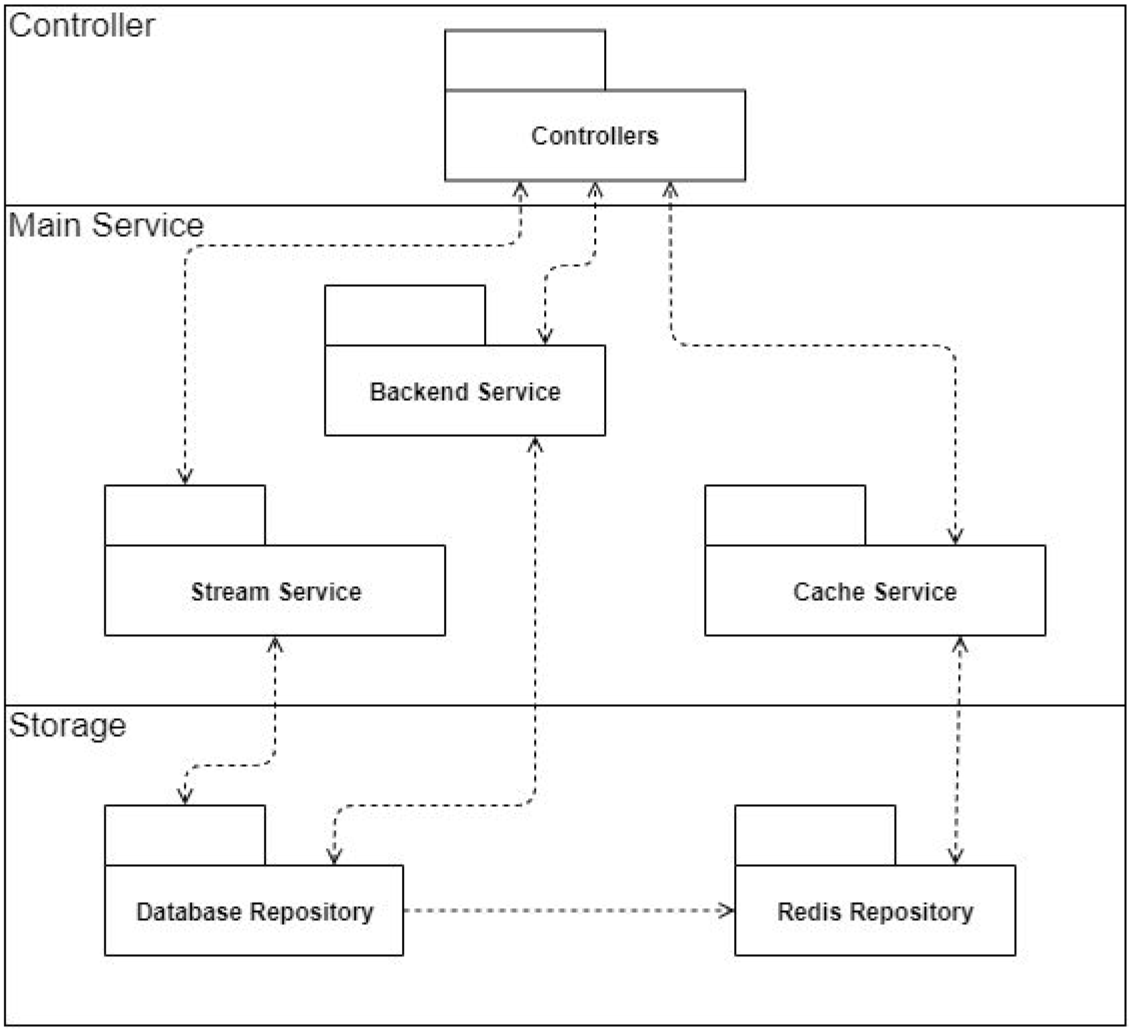

Figure 2: Subsystem Decomposition of the Backend System

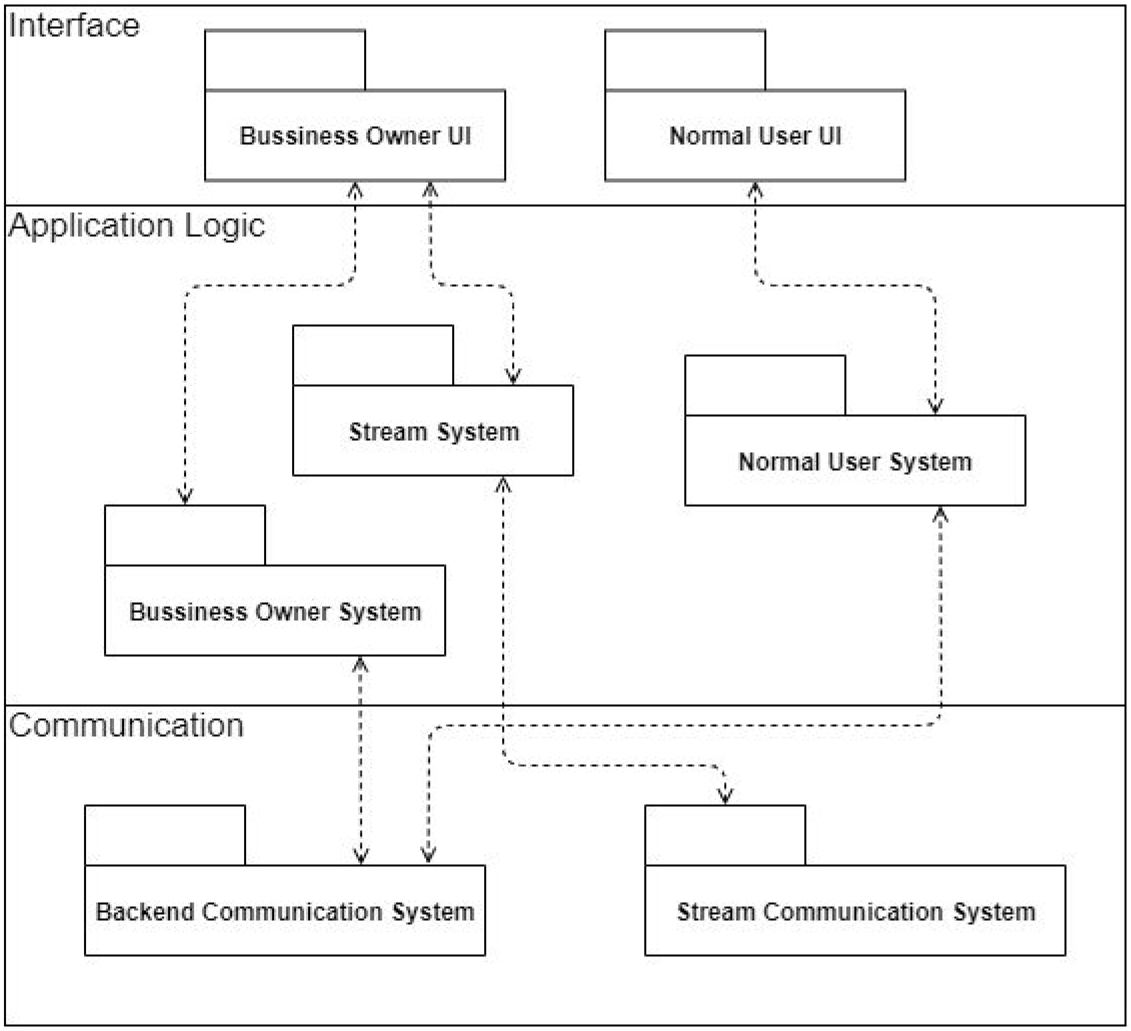

Figure 3: Subsystem Decomposition of the Mobile Application

2.1 Mobile Packages

The mobile application has its own programming perspective. This means that it will be programmed in a separate manner from the system which will only use the data provided from the backend via the database. The mobile application consists of two subsystems: the UI (user interface) and the Logic subsystem. The UI package is for the mobile applications' front-end side development. It will be used with the Logic subsystem. The logic package is for fetching the data from the backend and preparing the data for the UI subsystem. For instance, the logic subsystem will fetch rates for the mask and social distancing for every 15 minutes. For the past data graphs, it will store the arrived data in an array. It will make the information ready to be shown as a graph which will be shown by the UI subsystem. For mobile application JavaScript language will be used via TypeScript. Current web technologies, which also work on mobile which are built-on and for Javascript. However JavaScript lacks types; thus, we choose to use TypeScript which is a strongly-typed language and it compiles to JavaScript which is then run on the production environment. React Native is used as the framework for the user interface programming of the mobile application.

2.1.1 Mobile UI

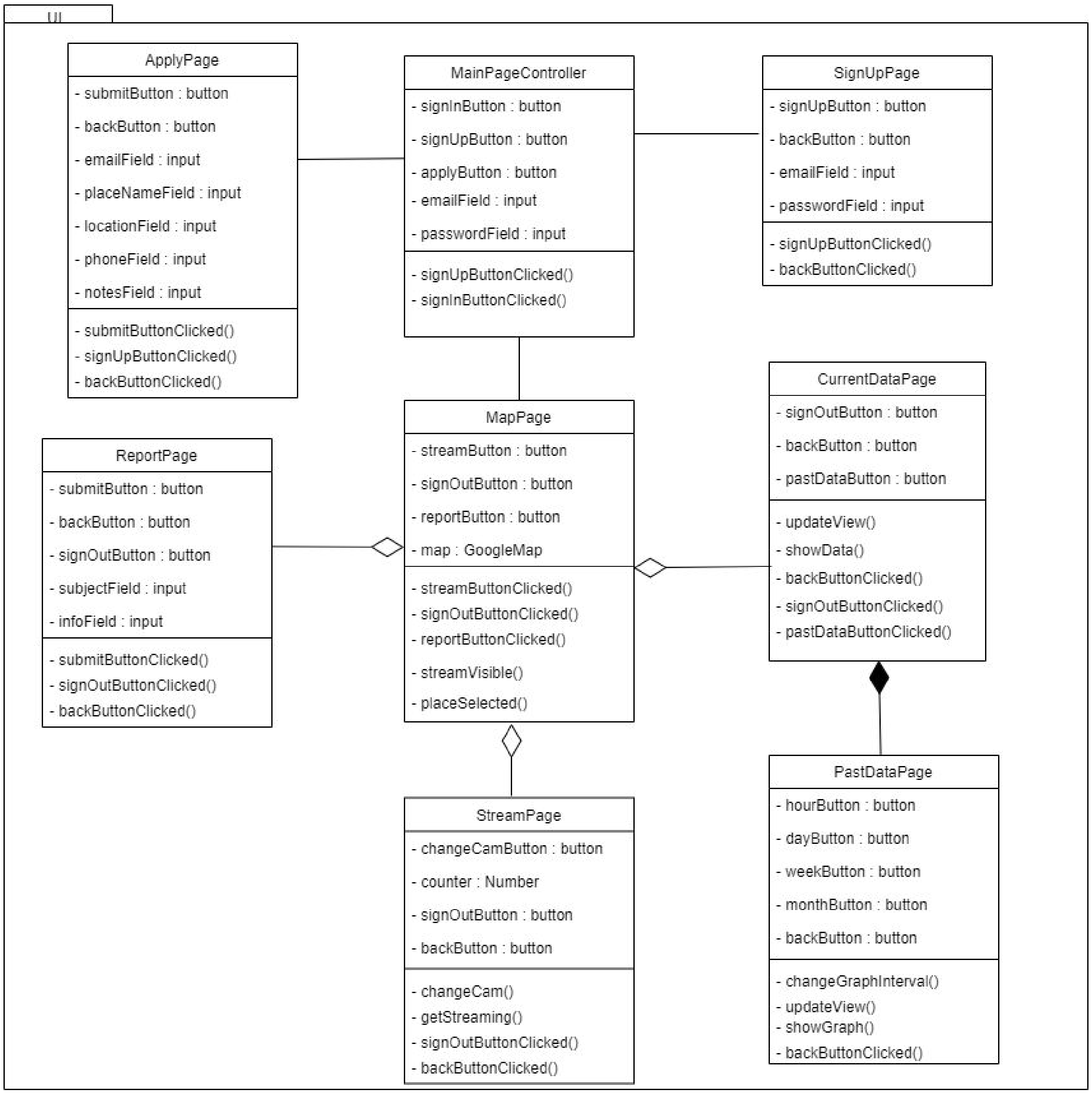

Figure 4. UI Package for Mobile Application

In this package, there are eight classes which are responsible for the user interface of the mobile application. MapPage is the page that contains the map that shows the places registered in the application as well as the report and streaming buttons. The MainPageController is the first page that the users see when they enter the application which contains the sign in, sign up buttons and related text fields. SignUpPage consists of the related sign up UI items. ApplyPage is for the users who want to sign up to the application as a place owner and contains related text fields and a submit button. ReportPage enables users to report the bugs and suggestions and it contains text fields and a submit button. StreamPage enables place owners to watch their own camera streams. CurrentDataPage and PastDataPage displays the current and past data UI items of the places, respectively.

2.1.2 Mobile Logic

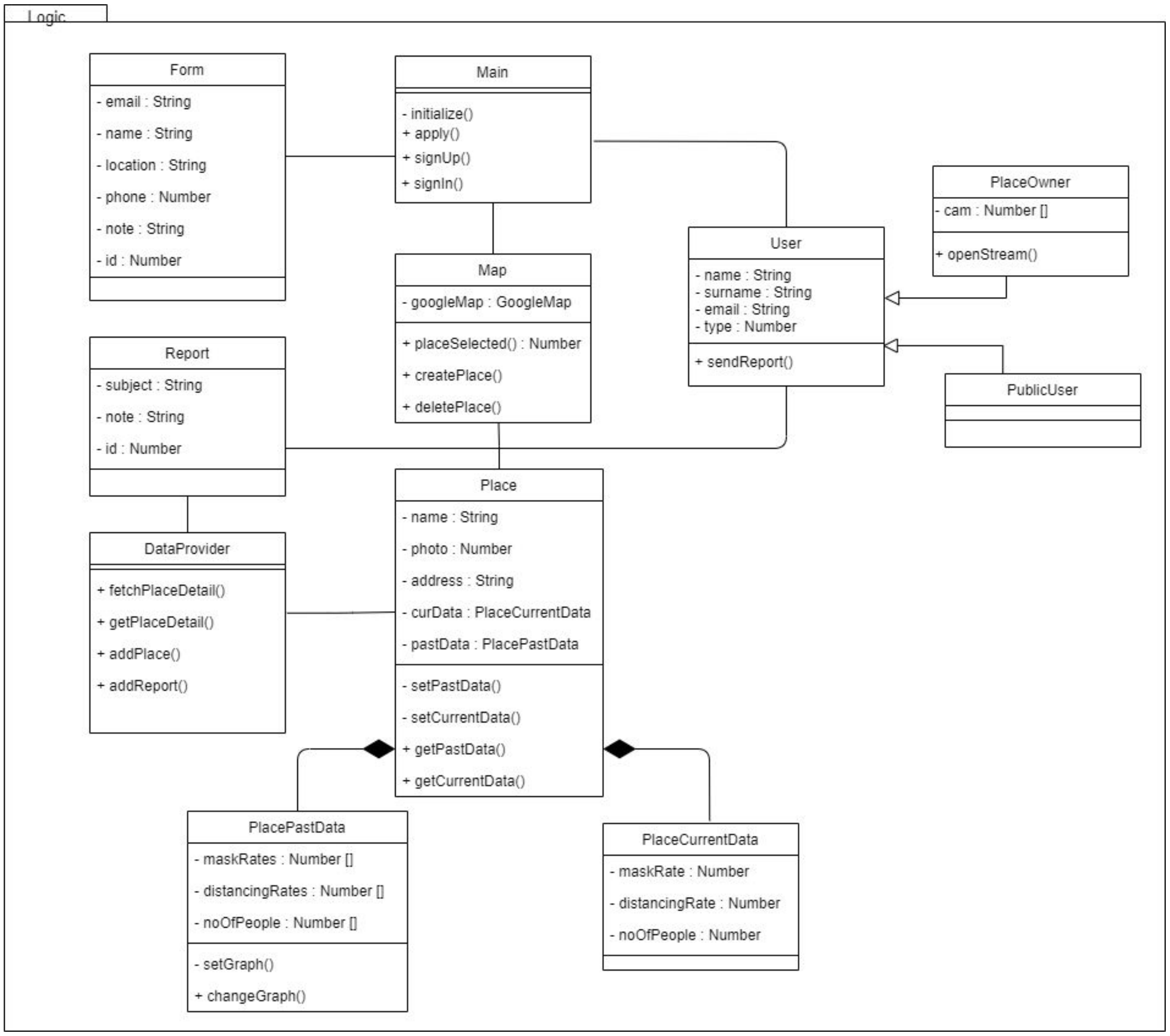

Figure 5. Logic Package for Mobile Application

In this package, there are eleven classes responsible for the logic of the mobile application. Main class initializes the application and it contains methods to sign in, sign up and apply. Map class does the map activities such as creating, deleting or selecting a place. Place class corresponds to the places registered in the application and displayed in the map. Place also contains current and past data. PlaceCurrentData and PlacePastData contain the compliance rates to the mask usage and social distance as well as the number of people in that place. There are two types of users in the application which are place owner and public user. The place owner has also streaming activity in addition to the public user. Form is required to sign up to the system for the place owners and Report is for the bugs and suggestions. DataProvider gets data from the database or changes according to the new data.

2.2 Backend Packages

2.2.1 Models

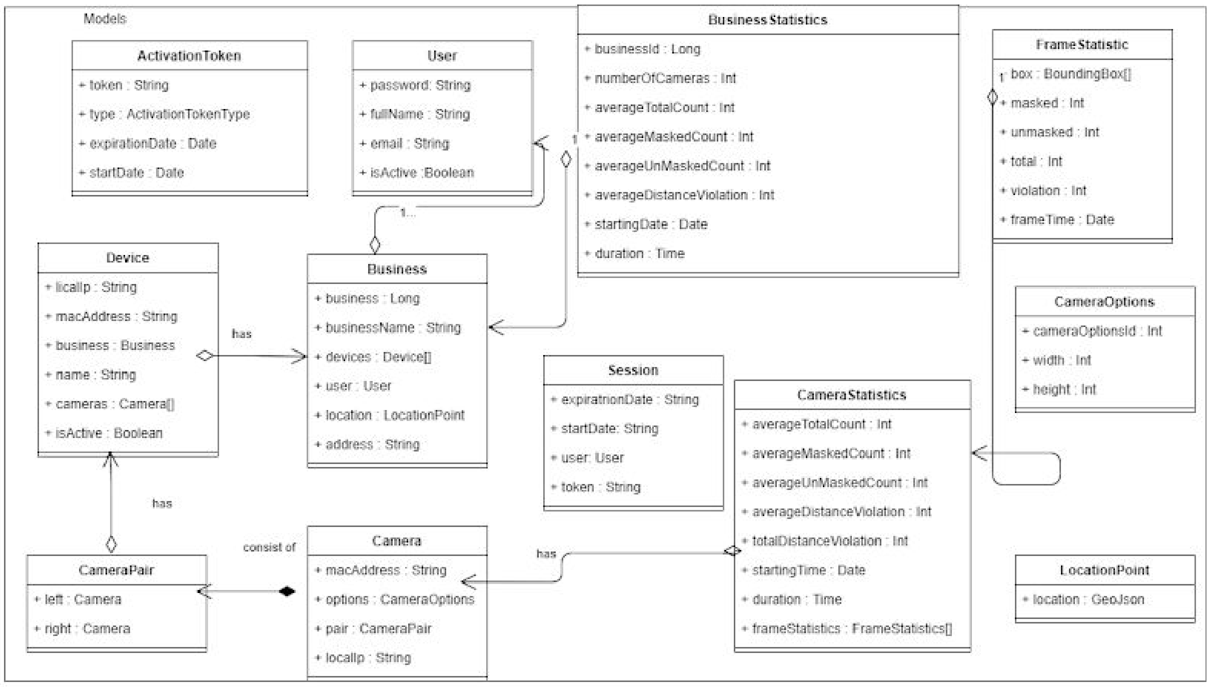

Figure 6. Models Package for Backend

That is the main model system. Here it can be seen that every user that connects with devices has one or more businesses. Each business has a device which is an interface on PC to control and check the camera streams. Two cameras generate a camera Pair which generates a better approximation of the depth map. With these camerapirs we can generate statistics that will be shared with our business owner and the other users who use this system on mobile.

2.2.2 Services

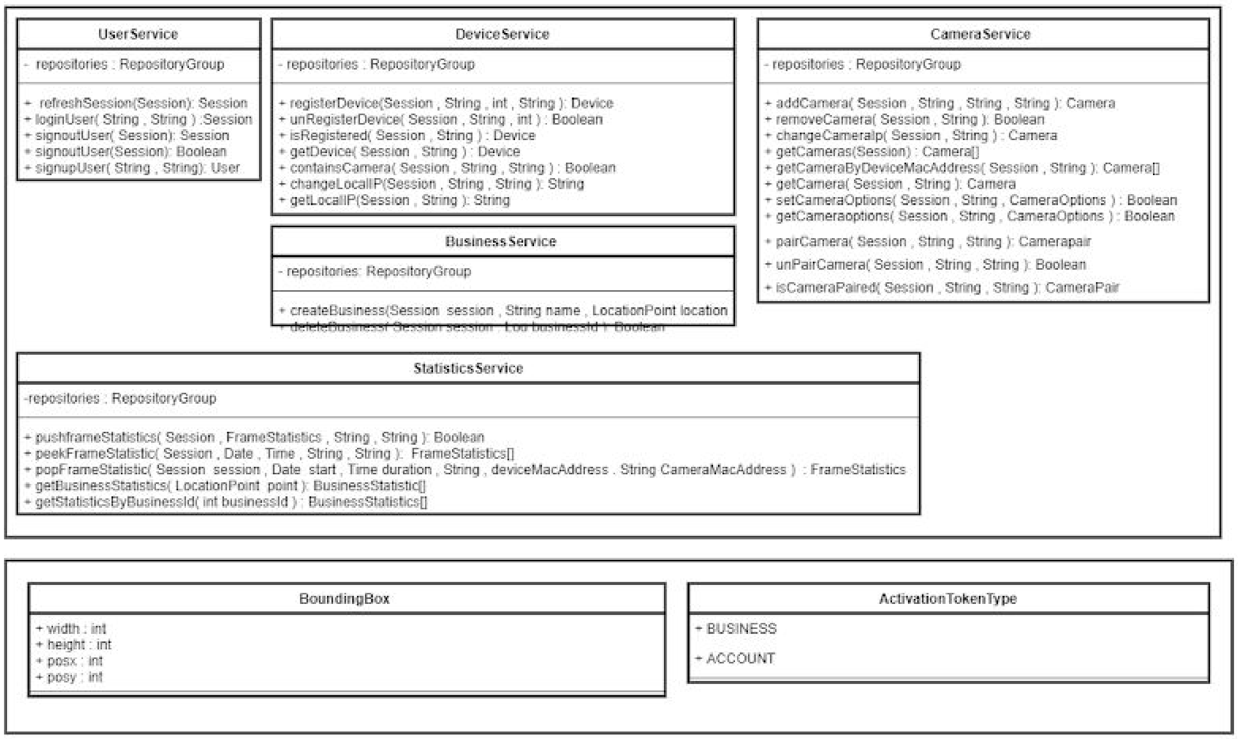
UserService
- repositories : RepositoryGroup
+ refreshSession(Session): Session
+ loginUser(String , String) :Session
+ signoutUser(Session): Session
+ signoutUser(Session): Boolean
+ signupUser(String , String): User
DeviceService
- repositories : RepositoryGroup
+ registerDevice(Session , String , int , String): Device
+ unRegisterDevice(Session , String , int) : Boolean
+ isRegistered(Session , String) : Device
+ getDevice(Session , String) : Device
+ containsCamera(Session , String , String) : Boolean
+ changeLocalIP(Session , String , String): String
+ getLocalIP(Session , String): String
CameraService
- repositories : RepositoryGroup
+ addCamera(Session , String , String , String): Camera
+ removeCamera(Session , String): Boolean
+ changeCameraIp(Session , String) : Camera
+ getCameras(Session) : Camera[]
+ getCameraByDeviceMacAddress(Session , String): Camera[]
+ getCamera(Session , String): Camera
+ setCameraOptions(Session , String , CameraOptions) : Boolean
+ getCameraoptions(Session , String , CameraOptions) : Boolean
+ pairCamera(Session , String , String): Camerapair
+ unPairCamera(Session , String , String): Boolean
+ isCameraPaired(Session , String , String): CameraPair
Business Service
- repositories: RepositoryGroup
+ createBusiness(Session session , String name , LocationPoint location
+ deleteBusiness(Session session , Lon businessId): Boolean
StatisticsService
-repositories : RepositoryGroup
+ pushframeStatistics(Session , FrameStatistics , String , String): Boolean
+ peekFrameStatistic(Session , Date , Time , String , String): FrameStatistics[]
+ popFrameStatistic(Session session , Date start , Time duration , String , deviceMacAddress . String CameraMacAddress) : FrameStatistics
+ getBusinessStatistics(LocationPoint point): BusinessStatistic[]
+ getStatisticsByBusinessId(int businessId) : BusinessStatistics[]
BoundingBox
+ width : int
+ height : int
+ posx : int
+ posy : int
ActivationTokenType
+ BUSINESS
+ ACCOUNT

Figure 7. Services Package for Backend

This subsystem contains the services offered by the backend system. These services are grouped and divided into classes that are responsible for one general subject. For example, UserService has all the functions for user related operations. Additionally, each service has a RepositoryGroup for accessing and calling repository functions that are located in the lower layer.

2.2.3 Repositories

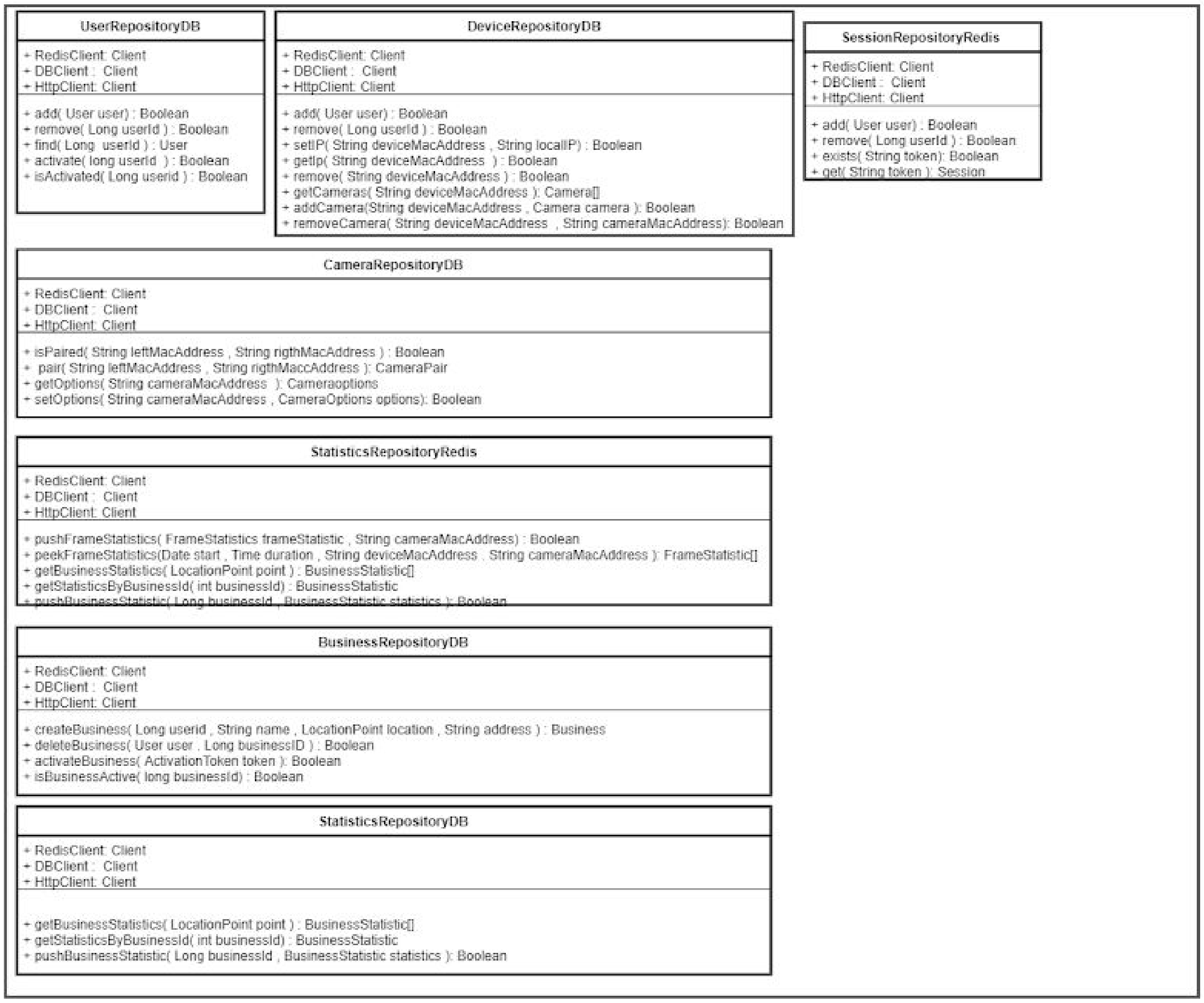
UserRepositoryDB
+ RedisClient: Client
+ DBClient : Client
+ HttpClient: Client
+ add(User user) : Boolean
+ remove(Long userId) : Boolean
+ find(Long userId) : User
+ activate(long userId) : Boolean
+ isActivated(Long userid) : Boolean
DeviceRepositoryDB
+ RedisClient: Client
+ DBClient : Client
+ HttpClient: Client
+ add(User user) : Boolean
+ remove(Long userId) : Boolean
+ setIP(String deviceMacAddress , String localIP) : Boolean
+ getIp(String deviceMacAddress) : Boolean
+ remove(String deviceMacAddress) : Boolean
+ getCameras(String deviceMacAddress): Camera[]
+ addCamera(String deviceMacAddress , Camera camera): Boolean
+ removeCamera(String deviceMacAddress , String cameraMacAddress): Boolean
SessionRepositoryRedis
+ RedisClient: Client
+ DBClient : Client
+ HttpClient: Client
+ add(User user) : Boolean
+ remove(Long userId) : Boolean
+ exists(String token): Boolean
+ get(String token): Session
CameraRepositoryDB
+ RedisClient: Client
+ DBClient : Client
+ HttpClient: Client
+ isPaired(String leftMacAddress , String rigthMacAddress) : Boolean
+ pair(String leftMacAddress , String rigthMaccAddress): CameraPair
+ getOptions(String cameraMacAddress): Cameraoptions
+ setOptions(String cameraMacAddress , CameraOptions options): Boolean
StatisticsRepositoryRedis
+ RedisClient: Client
+ DBClient : Client
+ HttpClient: Client
+ pushFrameStatistics(FrameStatistics frameStatistic , String cameraMacAddress) : Boolean
+ peekFrameStatistics(Date start , Time duration , String deviceMacAddress , String cameraMacAddress): FrameStatistic[]
+ getBusinessStatistics(LocationPoint point) : BusinessStatistic[]
+ getStatisticsByBusinessId(int businessId) : BusinessStatistic
+ pushBusinessStatistic(Long businessId , BusinessStatistic statistics): Boolean
BusinessRepositoryDB
+ RedisClient: Client
+ DBClient : Client
+ HttpClient: Client
+ createBusiness(Long userid , String name , LocationPoint location , String address) : Business
+ deleteBusiness(User user , Long businessID) : Boolean
+ activateBusiness(ActivationToken token): Boolean
+ isBusinessActive(long businessId) : Boolean
StatisticsRepositoryDB
+ RedisClient: Client
+ DBClient : Client
+ HttpClient: Client
+ getBusinessStatistics(LocationPoint point) : BusinessStatistic[]
+ getStatisticsByBusinessId(int businessId) : BusinessStatistic
+ pushBusinessStatistic(Long businessId , BusinessStatistic statistics): Boolean

Figure 8. Repositories Package for Backend

These are the postgres and redis repositories which help us to cache and store the data. The Redis repositories are interfaces that help the developers cache the given data. Other kinds of databases are interfaces for the postgres sql that we will use to store long-term data.

2.2.4 Types

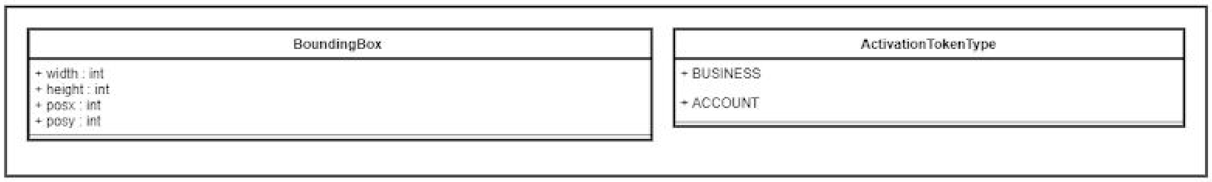

Figure 9. Enums for Ease of Use

2.3 Device Packages

2.3.1 Models

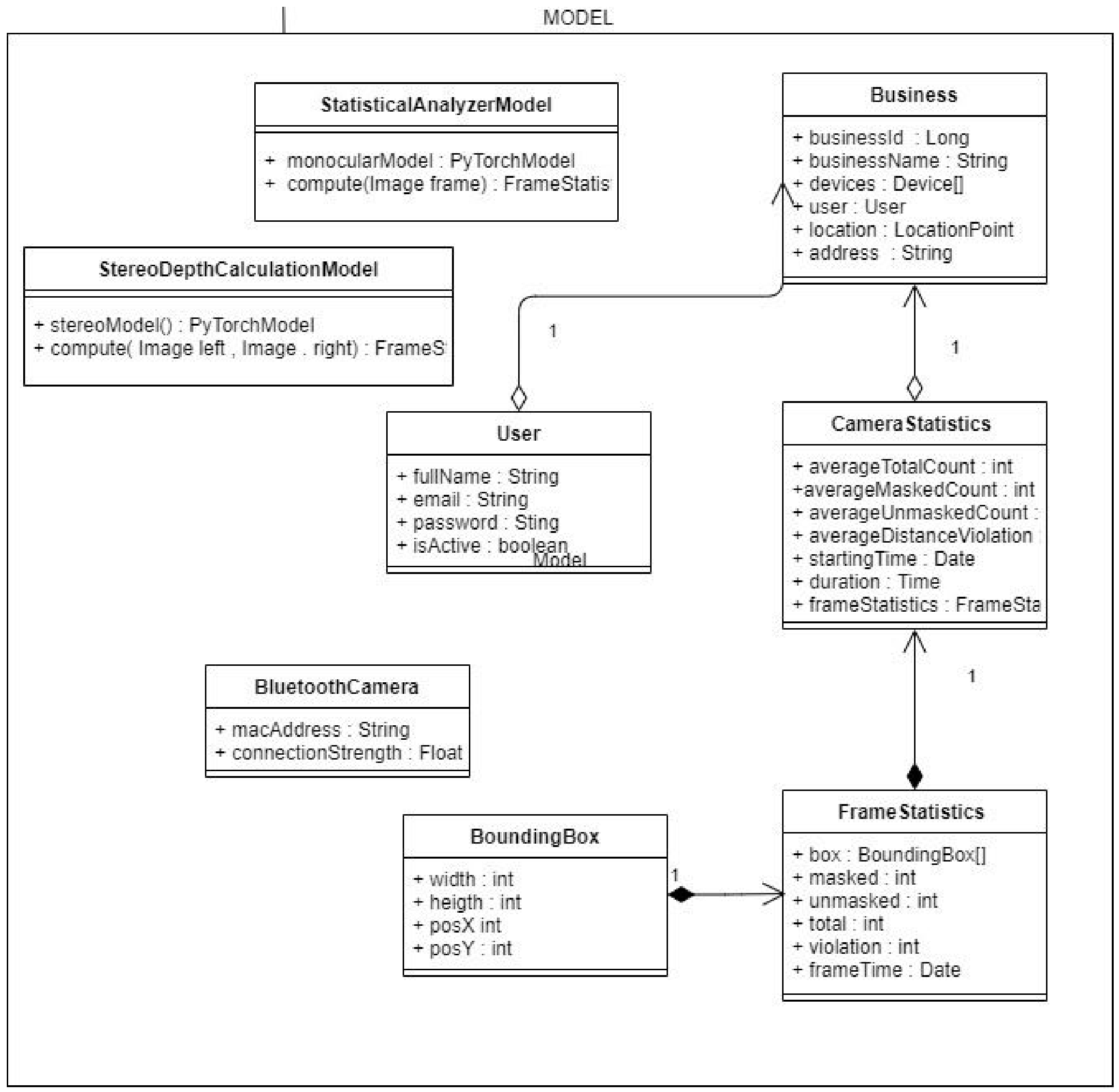

Figure 10. Models Package for the Device

These are the subsystems denoted for models located in the device. Some of those models are similar to the models on the backend service however, not all the models are transferred here. There are also device specific models such as BluetoothCamera.

2.3.2 Views

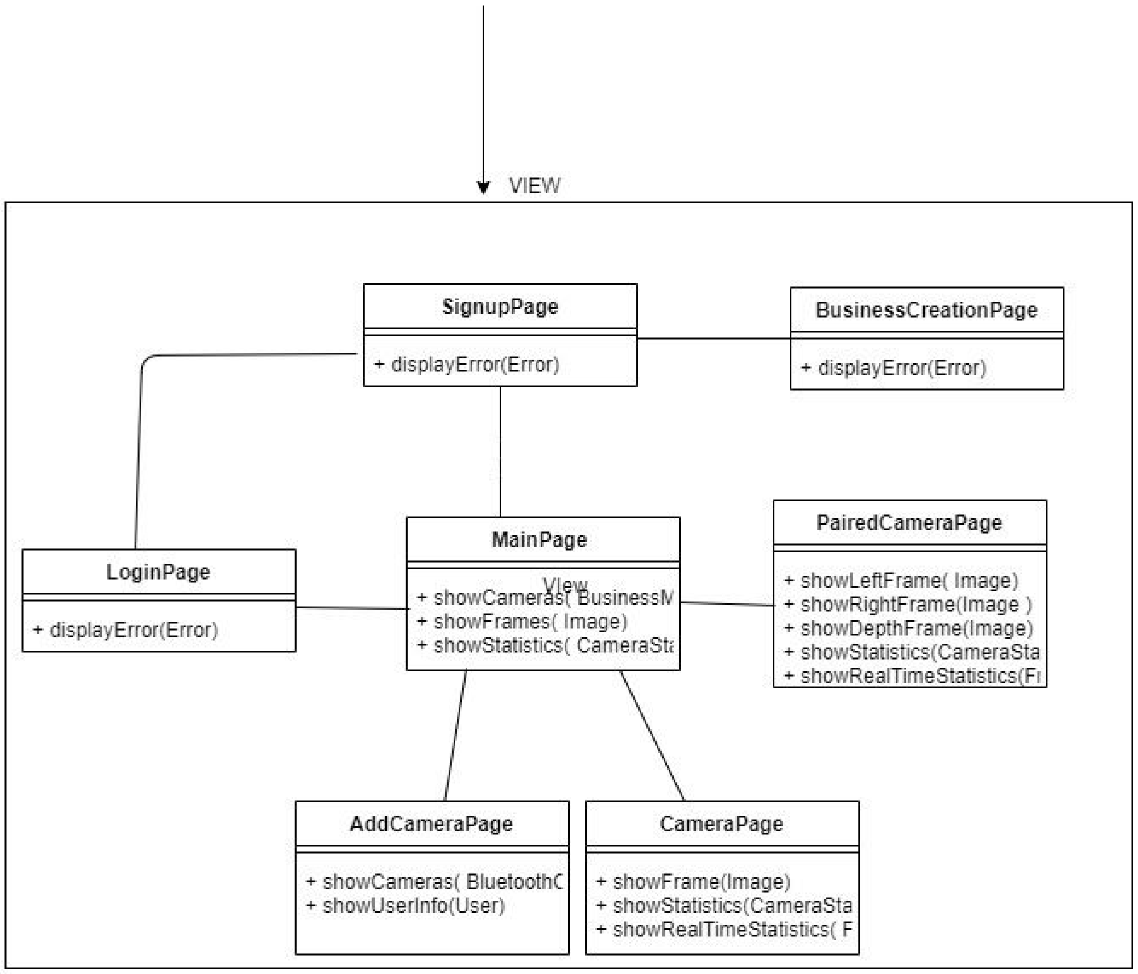

Figure 11. Views Package for Device

View part of the system. These classes contain parts in the interface of the system. This interface will only be in the Device, since we can see the camera stream here. The user can achieve the pages where he/she can watch her/his cameras, make them pair and add or remove cameras.

2.3.3 Controllers

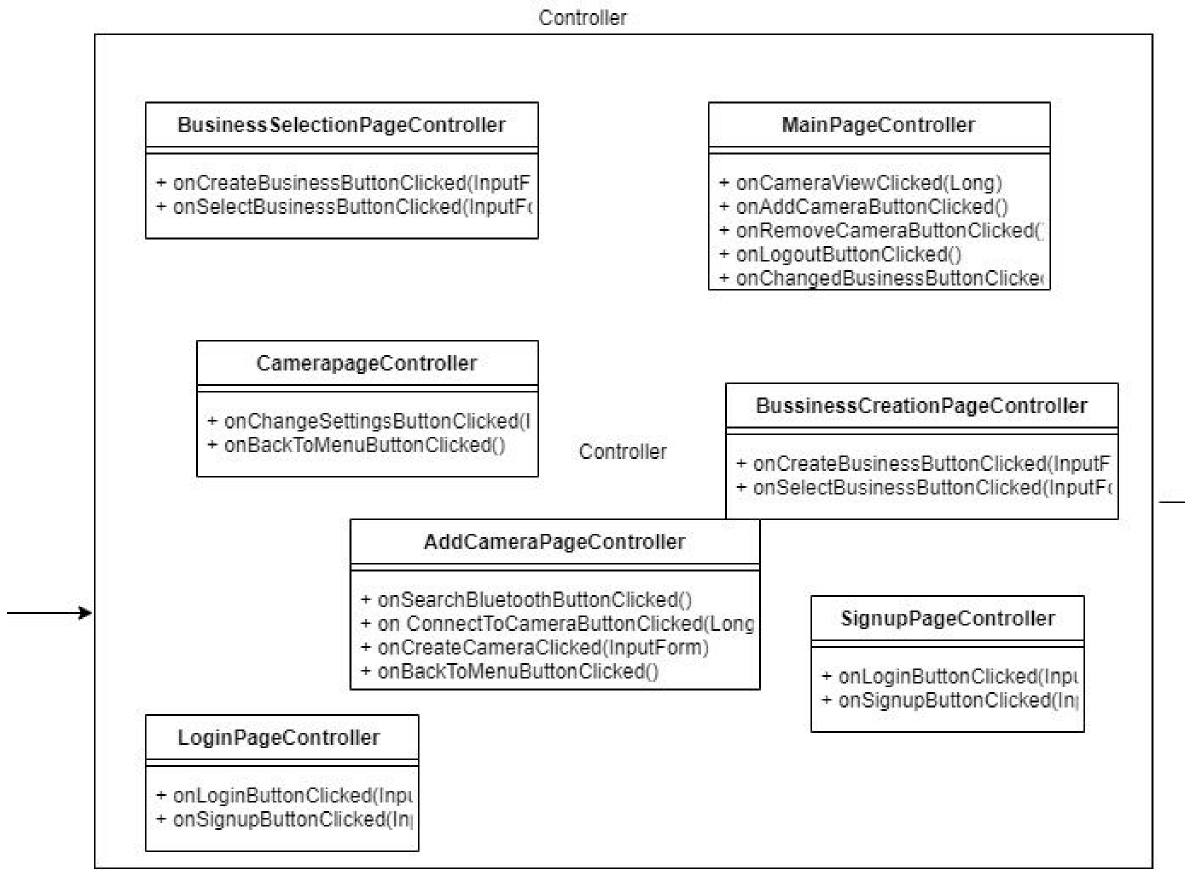
Controller
BusinessSelectionPageController
+ onCreateBusinessButtonClicked(InputF
+ onSelectBusinessButtonClicked(InputF
MainPageController
+ onCameraViewClicked(Long)
+ onAddCameraButtonClicked()
+ onRemoveCameraButtonClicked(
+ onLogoutButtonClicked()
+ onChangedBusinessButtonClicke
CamerapageController
+ onChangeSettingsButtonClicked(I
+ onBackToMenuButtonClicked()
Controller
BussinessCreationPageController
+ onCreateBusinessButtonClicked(InputF
+ onSelectBusinessButtonClicked(InputF
AddCameraPageController
+ onSearchBluetoothButtonClicked()
+ on ConnectToCameraButtonClicked(Long
+ onCreateCameraClicked(InputForm)
+ onBackToMenuButtonClicked()
SignupPageController
+ onLoginButtonClicked(Inpu
+ onSignupButtonClicked(In
LoginPageController
+ onLoginButtonClicked(Inpu
+ onSignupButtonClicked(In

Figure 12. Controllers Package for Device

This part of the system manipulates the data between model part and the view part. Every action taken by the view part will be sent here. And for each class in the view there is a corresponding controller for it. After that it does logical operations on the model part and sends it back to view.

3. Class Interfaces

3.1 Mobile Packages

3.1.1 Mobile UI

Class	MainPageController
	Class for the first page that will be shown when the app started.
Properties	
signInButton	Button - sign in button
signUpButton	Button - sign up button
applyButton	Button - apply button for those who want to be place owner
emailField	Input field - email field to get input
passwordField	Input field - password field to get input
Methods	
signUpButtonClicked()	Method - it directs the user to the SignUpPage.
signInButtonClicked()	Method - it checks whether the user's email and password that the user entered are valid, if so, it directs the user to the MapPage, otherwise, a warning is displayed.

Class	MapPage
	Class for the page that will contain the map, as well as the stream, report and sign out buttons.
Properties	
streamButton	Button to watch the stream of the place owner's own camera.
signOutButton	Button to sign out from the application.

reportButton	Button to report the bugs and suggestions to the team.
map	Google Map that will display the places on the map.
Methods	
streamButtonClicked()	Method that will direct the users to the StreamPage.
signOutButtonClicked()	Method that will sign out the users from the application.
reportButtonClicked()	Method that will direct the users to the ReportPage.
streamVisible()	Method that will check whether the user is a place owner or not, to make the stream button visible in the MapPage.
placeSelected()	Method that will get the place that the user has selected from the map.

Class	StreamPage
	Class for the streaming page that will be only shown to place owners.
Properties	
changeCamButton	Button - button for changing the cam for streaming view
signOutButton	Button - sign out button
backButton	Button - back button
counter	Number - It will be a variable within the file that counts the number of times pressed to the change cam button. When the limit of the number of cams is reached for that place owner it will be set to 0. It

	will be updated within the methods as the button is pressed.
Methods	
changeCam(Number counter)	It changes the streaming view when the change cam button is clicked.
getStreaming()	It shows the streaming view.
signOutButtonClicked()	Signs out the user, then opens the main page.
backButtonClicked()	Opens the previous page.

Class	CurrentDataPage
	Class for showing the current data of the selected place from the map.
Properties	
pastDataButton	Button - button for changing the cam for streaming view
signOutButton	Button - sign out button
backButton	Button - back button
Methods	
updateView()	Updates the current data view when new data is available.
showData()	Shows the data view in an organized manner.
backButtonClicked()	Opens the previous page.
signOutButtonClicked()	Signs out the user, then opens the main page.
pastDataButttonClicked()	It opens the page for detailed past data via PastDataPage class.

Class	PastDataPage

	Class to show the past data of the selected place from the map.
Properties	
hourButton	Button to show the last hour’s past data.
dayButton	Button to show the last day’s past data.
weekButton	Button to show the last week’s past data.
monthButton	Button to show the last month’s past data.
backButton	Button to go back to the MapPage.
Methods	
changeGraphInterval(Button id)	Method that changes the interval of the graph according to the user’s choice and adapts the graph according to the data of the hour, day, week or month.
updateView()	Method that updates the past data view when the new data is available.
showGraph()	Method that displays the graph.
backButtonClicked()	Method that will direct the user to the previous page.
placeSelected()	Method that will get the place that the user has selected from the map.

Class	ReportPage
	Class for the report page. It enables users to report bugs or to make any comments about our system.
Properties	
submitButton	Button - button for submitting the report form
signOutButton	Button - sign out button
backButton	Button - back button

subjectField	Input - Text field to get report subject from the user.
infoField	Input - Text field to get information about user's report.
Methods	
submitButtonClicked()	It submits the report form via using the logic package.
signOutButtonClicked()	Signs out the user, then opens the main page.
backButtonClicked()	Opens the previous page.

Class	ApplyPage
	Class that enables users to apply to the application to become a place owner.
Properties	
submitButton	Button to submit the application form.
backButton	Button to go back to the previous page.
emailField	Input field to get the user's email.
placeNameField	Input field to get the place's name.
locationField	Input field to get the place's location.
phoneField	Input field to get the place's phone.
notesField	Input field to get the user's notes.
Methods	
submitButtonClicked()	Method that submits the report.
signUpButtonClicked()	Method that will direct the user to the SignUpPage.
backButtonClicked()	Method that will direct the user to the previous page.

Class	SignUpPage

	Class that enables users to sign up to the application
Properties	
signUpButton	Button to sign up to the application.
backButton	Button to go back to the previous page.
emailField	Input field to get the user’s email.
passwordField	Input field to get the user’s password.
Methods	
signUpButtonClicked()	Method that will direct the user to the MainPageController to sign in.
backButtonClicked()	Method that will direct the user to the previous page.

3.1.2 Mobile Logic

Class	Main
	Main class for the logic subsystem to perform vital operations.
Methods	
initialize()	Initializes the system for the initial state.
apply()	Will be called by the UI subsystem to perform apply operation on the backend side.
signUp()	It communicates with the database system so that the signUp operation can be made. It checks whether the user already registered to the system, if not it adds to the database.
signIn()	Performs sign in operation.

Class	Map
	Class that uses GoogleMaps API to set the Map information so that UI can be arranged accordingly.
Properties	
googleMap	GoogleMap - Google map via API.
Methods	
placeSelected()	It returns the selected place's location, from the map.
createPlace()	Creates new place objects to be later shown on the map. Sets the new place information.
deletePlace()	Deletes a place object when needed.

Class	Place
	Class for place information.
Properties	
name	String - Name of a place
photo	Number - Photo of a place, as in form of base64 encoding.
address	String - Address of the place as text to be shown for the users.
curData	Object for PlaceCurrentData. UI will use this object with its information to arrange corresponding UI components.
pastData	Object for PlacePastData. UI will use this object with its information to arrange corresponding UI components.
Methods	

setPastData()	Creates and sets the pastData object.
setCurrentData()	Creates and sets the curData object
getPastData()	It will be used by the UI subsystem to fetch past data information for the view.
getCurrentData()	It will be used by the UI subsystem to fetch current data information for the view.

Class	PlacePastData
	Class for past data information of the places.
Properties	
maskRates	Number[] - Array that stores mask compliance rates for each 15 minutes interval data.
distancingRates	Number[] - Array that stores social distancing compliance rates for each 15 minutes interval data.
noOfPeople	Number[] - Array that stores the number of people for each 15 minutes interval data.
Methods	
setGraph()	Sets the graph information for initial opening.
changeGraph()	Arranges the graph information when the user wants to change the interval of the graph view.

Class	PlaceCurrentData
	Class for current data information for the places.
Properties	

maskRate	Number - stores mask compliance rate for the last 15 minute
distancingRate	Number - stores social distancing compliance rate for the last 15 minute
noOfPeople	Number - stores number of people for the last 15 minute

Class	Form
	Class that stores form information when a user applies to become a place owner.
Properties	
email	String - email information
name	String - name information
location	String - location information
phone	Number - phone number
note	String - note information for users to add additional notes.
id	Number - the integer that the form is represented with.

Class	Report
	Class that represents the report that the users send their suggestions and bugs.
Properties	
subject	String - that the users enter the subject of the report.
note	String - that the users enter the note of the report.

id	Number - the integer that the report is represented with.

Class	DataProvider
	Class to achieve efficient and quick communication with the database.
Methods	
fetchPlaceDetail()	It fetches place details from the database.
getPlaceDetail()	This method will be called from the other classes to get and set the place objects.
addPlace()	It will create new place objects within the mobile application system.
addReport()	It adds the report information to the database.

Class	User
	Class that represents the user in the application.
Properties	
name	String - the name of the user
surname	String - the surname of the user
email	String - the email address of the user
type	Number - represents the type of the user. If it is 0, the user is a public user, if it is 1, the user is a place owner.
Methods	
sendReport()	Method to send the report to the team.

Class	PlaceOwner
	Class that represents the user type place owner in the application.
Properties	
cam	Number[] - represents the cameras of the place owner
Methods	
openStream()	Method to watch the place owner’s own stream.

Class	PublicUser
	Class that represents the user type public user in the application.

3.2 Backend Packages

3.2.1 Models

Class	User
	This class models the user.
Properties	
public String fullName	Name of the user.
public String email	Email address of the user.
public String password	Hashed password of the user.
public Boolean isActive	Boolean variable denoting if a user is activated or not.
Methods	

Class	Session
	This class models the session.
Properties	
public Date expirationDate	Expiration date of the token.
public Date startDate	Creation date of the session.
public User user	User object that the session is pointing to.
public String token	JWT Token used for authentication.
Methods	

Class	Device
	This class models the device.
Properties	
public String localIp	Local IP address of the device.
public String macAddress	Mac Address of the device.
public Business business	Business object of the device.
public String name	Name of the device.
public Camera[] cameras	Cameras registered to the device as a list.
public Boolean isActive	The activity status of the device.
Methods	

Class	FrameStatistic
	This class models a FrameStatistic.
Properties	
public BoundingBox[] box	List of bounding boxes in the frame.
public Int masked	Number of masked people.
public Int unmasked	Number of unmasked people.
public Int total	Total number of people passed through cameras.
public Int violation	Number of people violated the rules
public Date frameTime	Time stamp of the frame.
Methods	

Class	Camera
	This class models the user.
Properties	
public String macAddres	Mac address of the camera.
public CameraOptions options	Set options of the camera.
public CameraPair pair	Representation of a camera pair of two cameras.
public String localIp	Local IP address of the camera.
Methods	

Class	CameraPair

	This class models the user.
Properties	
public Camera left	Camera that is on the left in the pair.
public Camera right	Camera that is on the right in the pair.
Methods	

Class	CameraOptions
	This class models the user.
Properties	
public Int cameraId	ID of the camera.
public Int width	Width in pixels of the camera.
public Int height	Height in pixels of the camera.
Methods	

Class	Business
	This class models the user.
Properties	
public Long businessId	Primary key, id of the business.
public String businessName	Name of the business.
public Device[] devices	List of devices registered to business
public User user	User object, owner of the business.
public LocationPoint location	Location of the business on the map.

public String address	Full physical address of the business.
Methods	

Class	BusinessStatistic
	This class models the business statistics.
Properties	
public Long businessId	ID stating which business the statistics belongs to.
public Int numberOfCameras	Number of cameras.
public Int averageTotalCount	Average count of people passing over time.
public Int averageMaskedCount	Average count of people with masks over time.
public Int averageUnMaskCount	Average count of people without masks over time.
public Int averageDistanceViolation	*Average count of people violating the distance rule over time.*
public Int totalDistanceViolation	Total count of violations of the distance rule.
public Date startingTime	Start time of recording,
public Time duration	Duration of the recording.
public CameraStatistics[] cameraStatistics	List of statistics in case of multiple cameras.
Methods	

Class	CameraStatistics
	This class models the user.

Properties	
public Int averageTotalCount	Average count of people passing over time.
public Int averageMaskedCount	Average count of people with masks over time.
public Int averageUnMaskCount	Average count of people without masks over time.
public Int averageDistanceViolation	Average count of people violating the distance rule over time.
public Int totalDistanceViolation	Total count of violations of the distance rule.
public Date startingTime	Start time of the recording.
public Time duration	Duration of the recording.
public FrameStatistic[] frameStatistics	List of frame statistics that composes camera statistics.

Class	LocationPoint
	This class models the Location Point.
Properties	
public GeoJson location	Custom JSON object for location.
Methods	

Class	ActivationToken
	This class models the activation token.
Properties	
public String token	Token itself.
public ActivationTokenType type	Type of the token.
public Date expirationDate	Expiration date of token.

public Date startDate	Start date of the token.
Methods	

3.2.2 Services

Class	UserService
	This class contains the core logic of user operations.
Properties	
private RepositoryGroup repositories	All repositories are accessed from this object.
Methods	
public Session refreshSession(Session session)	Extends the user’s session.
public Session loginUser(String email, String password)	Creates a session for the user.
public Boolean signoutUser(Session session)	Destroys the session for the user.
public User signupUser(String email. String password)	Registers the user but does not activate it. Also, creates an activation token for later activation.
public User activateUser(ActivationToken token)	Activates the user’s account.
public Boolean isUserActive(Session session)	Checks if the user is activated.

Class	DeviceService
	This class contains the core logic of device operations.
Properties	

private RepositoryGroup repositories	All repositories are accessed from this object.
Methods	
public Device registerDevice(Session session, String macAddress, Int businessId, String localIp)	Registers a device to the system.
public Boolean unRegisterDevice(Session session, String macAddress, Int businessId)	Deletes a device from the system.
public Device isRegistered(Session session, String macAddress)	Checks if a device is registered to the system.
public Device getDevice(Session session, String macAddress)	Gets the device using its mac address.
public Boolean containsCamera(Session session, String cameraMacAddress, String deviceMacAddress)	Checks if a device has a camera by the given mac address.
public String changeLocalIp(Session session, String macAddress, String localIp)	Changes a device’s local ip address
public String getLocalIp(Session session, String macAddress)	Gets a device’s local ip address.

Class	CameraService
	This class contains the core logic of camera operations.
Properties	
private RepositoryGroup repositories	All repositories are accessed from this object.
Methods	
public Camera addCamera(Session session, String deviceMacAddress, String cameraMacAddres, String ip)	Registers a camera.
public Boolean removeCamera(Session session, String cameraMacAddress)	Deletes a camera from the system.

public Camera changeCameraIp(Session session, String ip)	Changes the local ip address of a camera.
public Camera[] getCameras(Session session)	Get cameras of the user.
public Camera[] getCamerasByDeviceMacAddress(Session session, String deviceMacAddress)	Gets cameras registered to a device.
public Camera getCamera(Session session, String camMacAddress)	Gets the camera registered to a device.
public Boolean setCameraOptions(Session session, String cameraMacAddress, CameraOptions options)	Alters the camera options.
public Boolean getCameraOptions(Session session, String cameraMacAddress, CameraOptions options)	Gets the camera options.
public CameraPair pairCamera(Session session, String rightMacAddress, String leftMacAddress)	Pairs two cameras. If cameras are already paired it returns null.
public Boolean unPairCamera(Session session, String rightMacAddress, String leftMacAddress)	Unpairs two cameras. If cameras are already paired it returns false.
public CameraPair isCameraPaired(Session session, String rightMacAddress, String leftMacAddress)	Checks if given cameras are paired.

Class	StatisticsService
	This class contains the core logic of statistics operations.
Properties	
private RepositoryGroup repositories	All repositories are accessed from this object.
Methods	

public Boolean pushFrameStatistic(Session session, FrameStatistic frameStatistics, String deviceMAcAddress, String cameraMacAddress)	Pushes camera statistics to the system.
public FrameStatistic[] peekFrameStatistic(Session session, Date start, Time duration, String cameraMacAddress, String cameraMacAddress)	Returns but do not delete the frame statistics.
public FrameStatistic[] popFrameStatistic(Session session, Date start, Time duration, String deviceMacAddress, String cameraMacAddress)	Returns and deletes the frame statistics.
public BusinessStatistic[] getBusinesseStatistics(LocationPoint point)	Calculates the business statistics from the given point.
public BusinessStatistic[] getStatisticsByBusinessId(Int businessId)	Calculates the business statistics from the given business id.

Class	BusinessService
	This class contains the core logic of business operations.
Properties	
private RepositoryGroup repositories	All repositories are accessed from this object.
Methods	
public Business createBusiness(Session session, String name, LocationPoint location,String address)	Creates an unactivated business.
public Boolean deleteBusiness(Session session, Long businessId)	Deletes a business.
public Boolean activateBusiness(ActivationToken token)	Activates a business.

public Boolean isBusinessActive(Session session, Long businessId)	Checks if a business is active.

3.2.3 Types

Class	BoundingBox
	This class is a utility class that models a bounding box. The table of this entity won't be created.
Properties	
public Int width	Width of the bounding box in the pixel unit.
public Int height	Height of the bounding box in the pixel unit.
public Int posX	X position of the bounding box in the pixel unit.
public Int posY	Y position of the bounding box in the pixel unit.
Methods	

Enum	ActivationTokenType
	This class is a utility class.
Properties	
public BUSINESS	Indicated business token.
public ACCOUNT	Indicated account token.
Methods	

3.2.4 Repositories

Class	UserRepositoryDB
	This class is responsible for communicating with the database.
Properties	
public RedisClient client	The dependency supplied from the libraries to allow easy access to redis.
public DBClient client	The dependency supplied from the libraries to allow easy access to the database.
public HttpClient client	The dependency supplied from the libraries to allow easy web communication.
Methods	
public Boolean add(User user)	Creates an unactivated user.
public Boolean remove(Long userId)	Removes the user.
public User find(Long userId)	Finds and fetches the user.
public Boolean activate(Long userId)	Activates a user.
public Boolean isActivated(Long userId)	Checks if a user is activated.

Class	SessionRepositoryRedis
	This class is responsible for communicating with the Redis.
Properties	
public RedisClient client	The dependency supplied from the libraries to allow easy access to redis.
public DBClient client	The dependency supplied from the libraries to allow easy access to the database.
public HttpClient client	The dependency supplied from the libraries to allow easy web communication.

Methods	
public Session add(String token)	Creates a new session and inserts it into the Redis.
public Boolean remove(String token)	Removes a session from the Redis.
public Boolean exists(Sting token)	Check if the session exists.
public Session get(String token)	Get the Session from Redis.

Class	DeviceRepositoryDB
	This class is responsible for communicating with the database.
Properties	
public RedisClient client	The dependency supplied from the libraries to allow easy access to redis.
public DBClient client	The dependency supplied from the libraries to allow easy access to the database.
public HttpClient client	The dependency supplied from the libraries to allow easy web communication.
Methods	
public Device add(String deviceMacAddress)	Register a device to the database.
public Boolean remove(String deviceMacAddress)	Remove a device from the database.
public Boolean setIp(String deviceMacAddress, String localIp)	Change the ip address of the device.
public String getIp(String deviceMacAddress)	Get the ip address of the device.
public Boolean remove(String deviceMacAddress)	Remove the device.
public Camera[] getCameras(String deviceMacAddress)	Get the assigned cameras of a device.

public Boolean addCamera(String deviceMacAddress, Camera camera)	Add a camera to the device.
public Boolean removeCamera(String deviceMacAddress, String cameraMacAddress)	Remove the camera from the device.

Class	CameraRepositoryDB
	This class is responsible for communicating with the database.
Properties	
public RedisClient client	The dependency supplied from the libraries to allow easy access to redis.
public DBClient client	The dependency supplied from the libraries to allow easy access to the database.
public HttpClient client	The dependency supplied from the libraries to allow easy web communication.
Methods	
public Boolean isPaired(String leftMacAddress, String rightMacAddress)	Checks if a camera is paired.
public CameraPair pair(String leftMacAddress, String rightMacAddress)	Pairs given two cameras.
public CameraOptions getOptions(String cameraMacAddress)	Gets the options of a camera.
public Boolean setOptions(String cameraMacAddress, CameraOptions options)	Sets the options of a camera.

Class	StatisticsRepositoryRedis
	This class is responsible for communicating with the Redis.

Properties	
public RedisClient client	The dependency supplied from the libraries to allow easy access to redis.
public DBClient client	The dependency supplied from the libraries to allow easy access to the database.
public HttpClient client	The dependency supplied from the libraries to allow easy web communication.
Methods	
public Boolean pushFrameStatistic(FrameStatistic frameStatistic, String cameraMacAddress)	Pushes a camera statistic to the redis.
public FrameStatistic[] peekFrameStatistics(Date start, Time duration, String deviceMacAddress, String cameraMacAddress)	Gets but does not delete the FrameStatistics.
public FrameStatistic[] popFrameStatistics(Date start, Time duration, String deviceMacAddress, String cameraMacAddress)	Gets and deletes the FrameStatistic from the redis.
public BusinessStatistic[] getBusinesseStatistics(LocationPoint point)	Gets the business statistics from the database by location.
public BusinessStatistic getStatisticsByBusinessId(Int businessId)	Gets the business statistics from the database by business id.
public Boolean pushBusinessStatistic(Long businessId, BusinessStatistic statistics)	Pushes a business statistics to the redis.

Class	StatisticsRepositoryDB
	This class is responsible for communicating with the database.
Properties	
public RedisClient client	The dependency supplied from the libraries to allow easy access to redis.

public DBClient client	The dependency supplied from the libraries to allow easy access to the database.
public HttpClient client	The dependency supplied from the libraries to allow easy web communication.
Methods	
public BusinessStatistic[] getBusinesseStatistics(LocationPoint point)	Gets the business statistics from the database by location.
public BusinessStatistic getStatisticsByBusinessId(Int businessId)	Gets the business statistics from the database by business id.
public Boolean pushBusinessStatistic(Long businessId, BusinessStatistic statistics)	Pushes a business statistics to the database.

Class	BusinessRepositoryDB
	This class is responsible for communicating with the database.
Properties	
public RedisClient client	The dependency supplied from the libraries to allow easy access to redis.
public DBClient client	The dependency supplied from the libraries to allow easy access to the database.
public HttpClient client	The dependency supplied from the libraries to allow easy web communication.
Methods	
public Business createBusiness(Long userId, String name, LocationPoint location,String address)	Adds a business to the database.
public Boolean deleteBusiness(User user, Long businessId)	Deletes a business from the system.
public Boolean activateBusiness(ActivationToken token)	Activates a business.

public Boolean isBusinessActive(Long businessId)	Checks if business is active.

3.3 Device Packages

3.3.1 Models

Class	CameraStatistics
	This class models the user.
Properties	
public Int averageTotalCount	Return the average total count of the individuals in the frames
public Int averageMaskedCount	Return the average masked total count of the individuals in the frames
public Int averageUnMaskCount	Return the average unmasked total count of the individuals in the frames
public Int averageDistanceViolation	Return the average individuals who violates the social distance
public Int totalDistanceViolation	Return the total individuals who violates the social distance
public Date startingTime	Return the starting time Date
public Time duration	Time span of the statistics.
public FrameStatistic[] frameStatistics	List of FrameStatistics.
Methods	

Class	StatisticalAnalyzerModel
	This class models statistical analyzer.
Properties	

public PyTorchModel monocularModel	Pytorch convolutional neural network model.
Methods	
public FrameStatistics compute(Image frame)	Computes the FrameStatistics from a monocular image.

Class	StereoDepthCalculationModel
	This class models the stereo depth calculation.
Properties	
public PyTorchModel stereoModel	Pytorch convolutional neural network model.
Methods	
public FrameStatistics compute(Image left, Image right)	Computes the FrameStatistics from two images.

Class	BluetoothCamera
	This class models the user.
Properties	
public String macAddress	Mac address of the camera.
public Float connectionStrength	Connection strength of the camera to the device.
Methods	

Class	Business
	This class models the user.
Properties	

public Long businessId	Id of the business.
public String businessName	Name of the business.
public Device[] devices	List of devices of the
public User user	Owner user of the business.
public LocationPoint location	Location of the business.
public String address	Address of the business.
Methods	

Class	FrameStatistics
	This class models the user.
Properties	
public BoundingBox[] box	List of bounding boxes in the frame.
public Int masked	Number of masked people inside the frame.
public Int unmasked	Number of unmasked people inside the frame.
public Int total	Number of people inside the frame.
public Int violation	Number of violations in the frame.
public Date frameTime	The moment the frame has shot.
Methods	

Class	BoundingBox
	This class models the bounding box drawn around detected people.
Properties	

public Int width	Width of the bounding box in the pixel unit.
public Int height	Height of the bounding box in the pixel unit.
public Int posX	X position of the bounding box in the pixel unit.
public Int posY	Y position of the bounding box in the pixel unit.
Methods	

Class	CameraStatistics
	This class models statistics obtained from the camera feed.
Properties	
public Int averageTotalCount	Average number of people present inside the frame in the time span.
public Int averageMaskedCount	Average number of masked people inside the time span.
public Int averageUnMaskCount	Average number of unmasked people inside the time span.
public Int averageDistanceViolation	Average number of distance violations inside the time span.
public Int totalDistanceViolation	Total number of distance violations inside the time span.
public Date startingTime	Starting time of the statistics.
public Time duration	Time span of the statistics.
public FrameStatistic[] frameStatistics	List of FrameStatistic objects
Methods	

3.3.2 Views

Class	LoginPage
	This is the view for the login page.
Properties	
Methods	
public void displayError(Error error)	This method displays an error incase of an unsuccessful login.

Class	SignupPage
	This is the signup page view for users.
Properties	
Methods	
public void displayError(Error error)	This method displays an error incase of an unsuccessful signup.

Class	BusinessCreationPage
	This class acts as the view for the business creation process.
Properties	
Methods	
public void displayError(Error error)	This method displays an error incase of an unsuccessful business addition to the system.

Class	BusinessSelectionPage
	This is the view where the businesses are selected.
Properties	
Methods	
public void showBusiness(BusinessModel[] businesses)	This method takes BusinessModel as an input and displays it to the view.
public void displayError(Error error)	This method displays an error incase of an unsuccessful business addition to the system.

Class	MainPage
	This is the view for the main page.
Properties	
Methods	
public void showCameras(BluetoothCamera[] cameras)	Returns a list of available cameras.
public void showFrames(Image image)	Shows the image on the view.
public void showStatistics(CameraStatistics cameraStatistics)	Shows the statistics obtained from camera statistics on the view.

Class	AddCameraPage
	This is the view for adding cameras.
Properties	

Methods	
public void showCameras(BluetoothCamera[] cameras)	This method shows the available cameras on the view.
public void showUserInfo(User user)	This method shows information about the user on the view.

Class	CameraPage
	This is the view for the camera page.
Properties	
Methods	
public void showFrame(Image image)	This method displays a single frame on the view.
public void public void showStatistics(CameraStatistics cameraStatistics)	This method shows the statistics obtained by camera(s).
public void public void showRealTimeStatistics(FrameStatistic frameStatistics)	This method displays real time statistics on the view.

Class	PairedCameraPage
	This is the view for the main page.
Properties	
Methods	
public void showLeftFrame(Image image)	This method displays the frame coming from the left camera of the pair.
public void showRightFrame(Image image)	This method displays the frame coming from the right camera of the pair.

public void showDepthFrame(Image image)	This method displays the depth obtained by the pair of two cameras.
public void public void showStatistics(CameraStatistics cameraStatistics)	This method shows statistics obtained by the pair of cameras.
public void public void showRealTimeStatistics(FrameStatistic frameStatistics)	This method shows statistics of a frame obtained by the pair of cameras.

3.3.3 Controllers

Class	LoginPageController
	This is the controller for the user login page.
Properties	
Methods	
public void onLoginButtonClicked(InputForm form)	This method takes InputForm as an input and validates the login of a user.
public void onSignupButtonClicked(InputForm form)	This method takes an InputForm as an input and validates the signup of a user.

Class	SignupPageController
	This is the controller for the user signup page.
Properties	
Methods	
public void onLoginButtonClicked(InputForm form)	This method takes InputForm as an input and validates the login of a user.
public void onSignupButtonClicked(InputForm form)	This method takes an InputForm as an input and validates the signup of a user.

Class	BusinessCreationPageController
	This is the view for the registration of a business.
Properties	
Methods	
public void onCreateBusinessButtonClicked(InputForm form)	This method takes InputForm as an input and validates the business creation process.
public void onSelectBusinessButtonClicked(InputForm form)	This method takes InputForm as input from the click and displays the selected business.

Class	MainPageController
	This is the view for the main page.
Properties	
Methods	
public void onCameraViewClicked(Long id)	This method shows the view of the clicked camera by using its ID obtained from click.
public void onAddCameraButtonClicked()	This method handles adding the camera action.
public void onRemoveCameraButtonClicked()	This method removes a camera.
public void onLogoutButtonClicked()	This method logs out the user.
public void onChangeBusinessButtonClicked()	This method changes the business in view when clicked.

Class	AddCameraPageController
	This is the controller for adding cameras.
Properties	
Methods	
public void onSearchBluetoothButtonClicked()	This method searches for available bluetooth cameras.
public void onConnectToCameraButtonClicked(Long id)	This method connects to the selected camera.
public void onCreateCameraClicked(InputForm form)	This method takes InputForm as input and creates a camera.
public void onBackToMenuButtonClicked()	This method takes the user back to the Main Menu.

Class	CameraPageController
	This is the controller responsible for the camera page.
Properties	
Methods	
public void onChangeSettingsButtonClicked(InputForm form)	This method handles changing the settings of the camera.
public void onBackToMenuButtonClicked()	This method takes the user back to the main menu.

Class	PairedCameraPageController
	This is the controller responsible for the paired camera page.
Properties	
Methods	
public void onChangeSettingsButtonClicked(InputForm form)	This method handles changing the settings of the paired camera.
public void onBackToMenuButtonClicked()	This method takes the user back to the main menu.

4. Glossary

***React Native**:* It is an open-source mobile application framework in order to write code for Android, iOS and Windows which is developed by Facebook [1].

***OpenCV**:* (Open Source Computer Vision Library) is an open source computer vision and machine learning software library [2].

***Redis**:* (Remote Dictionary Server) is a data structure server in order to hold the data in the RAM. Redis is also an open-source NoSQL based database system [3].

***TypeScript**:* TypeScript is an open-source project backed by the tech giant Microsoft. Its basic premise is JavaScript with types. TypeScript compiles directly to JavaScript [4].

***ESP-32 Cam**:* The ESP32 is a low-cost system-on-chip (SoC) series that can be used in the development of IoT projects and embedded systems. It has Wi-Fi and Bluetooth capabilities that can provide many functionalities [5].

5. References

[1] “Learning React Native,” *O'Reilly Online Learning*. [Online]. Available: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html. [Accessed: 08-Feb-2021].

[2] “About,” *OpenCV*, 04-Nov-2020. [Online]. Available: https://opencv.org/about/. [Accessed: 08-Feb-2021].

[3] *Redis*. [Online]. Available: https://redis.io/. [Accessed: 08-Feb-2021].

[4] “Typed JavaScript at Any Scale.,” *TypeScript*. [Online]. Available: https://www.typescriptlang.org/. [Accessed: 08-Feb-2021].

[5] M. Schwartz, “Getting Started with the ESP32,” *Home*, 17-Nov-2020. [Online]. Available: https://makecademy.com/getting-started-esp32. [Accessed: 08-Feb-2021].