
Bilkent University
Department of Computer Engineering

# Senior Design Project

*Project short-name: EyeContact*

## Low-Level Design Report

Melisa Onaran – 21301232
Sarp Saatçioğlu – 21400375
Yunus Ölez – 21401539
Nazlı Abaz – 21400231

**Supervisor:** Hamdi Dibeklioğlu
**Jury Members:**
Çiğdem Gündüz Demir
Selim Aksoy
**Innovation Expert:** Çağla Çığ Karaman
**Website:** eyecontact.in

Feb 12, 2018

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 CS491/2.

# Table of Contents

# 1. Introduction

Improvement of every society depends on to the contribution of individuals in it. More people contributing to it with higher rates mean faster and bigger development. Governments endeavor to create better standards to increase production of individuals. However, individuals with disabilities do not have the same living conditions since they have special needs and thus, they cannot be as productive as others.

The aim of this project is to increase the living conditions of people who are visually impaired, and thus increase the development of societies. These individuals cannot get proper visually feedback from the environment which causes them to have a limited communication with societies. This project offers a solution for visually handicapped individuals that verbalizes the environment by using image processing using computer vision. Therefore, these people will be able to communicate, create and produce more.

This report is the transformation of the analysis model into a system design model. It defines design goals of the project, and decompose the system into smaller subsystems. Strategies for building the system, such as the hardware/software platform on which the system will run, the persistent data management strategy, global control flow, the access control policy, and the handling of boundary conditions are also mentioned in this report [1].

## 1.1. Object design trade-offs

- **Complexity vs. Functionality**

EyeContact is a social responsibility project such that it includes much useful functionalities for visually impaired users. It combines many software APIs and systems in order to provide better services and usage. Combination of those software APIs such as WebRTC, OpenCv and OpenFace increase the

complexity of the system. Therefore, it becomes harder to provide a stable functionality respectively.

- **Security vs. User Friendliness**

There is a great amount of data transaction between the application within the inner systems (mostly APIs) and database and also between outer system and database. In order to provide information efficiently, within the runtime of the application, to user in anywhere through an internet-connected desktop device, the database needs to be up and running every time. Hence, it makes system open to security threats. This is an important trade-off which will not be allowed to compromise from both sides.

- **Memory vs. Data Usage**

EyeContact has image processing algorithms and accordingly in order to perform this algorithm, we have an approach for both the client side and server side. This data processing causes that memory will be consumed highly by the processes. And it is mainly important to get the algorithm on time for visual and audio feedback.

- **Response Time vs. Performance**

EyeContact needs to have high productivity to be able to have a synchronous data flow. Since some complex image processing algorithms will be used in order to detect the faces and clothes with the high accuracy at the same time, our system finds that algorithm instantly and responses as fast as possible for the efficiency.

## 1.2. Interface documentation guidelines

In this document, the following convention is used:

1. Package name

   This indicates which class belongs to which package

2. Class name

   The specific class name which contains the following attributes and methods

3. Description

   A brief explanation of class

4. Attributes

   "+" indicates that the attribute is public and "-" indicates it is private. The type of the attribute is followed by ":".

5. Methods

   "+" indicates that the method is accessible from outside of the class and "-" indicates that it can be used only within the class. "#" sign indicates that the method is protected. Furthermore, their return types are specified after ":" sign.

## 1.3. Engineering standards

Unified Modeling Language (UML) is used in this report in order to demonstrate the structure of the system in a standard way. By this way, the diagrams become more comprehensible. For citations, IEEE style is used which is also a widely used engineering standard.

## 1.4. Definitions, acronyms, and abbreviations

- **WebSocket:** WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket API in Web IDL is being standardized by the W3C.
- **WebRTC:** WebRTC (Web Real-Time Communication) is a free, open-source project that provides web browsers and mobile applications with real-time communication (RTC) via simple application programming interfaces (APIs). It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps. Supported by Google, Mozilla, and Opera, WebRTC is being standardized through the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF).
- **Kurento:** Kurento is a WebRTC media server and a set of client APIs simplifying the development of advanced video applications for web and smartphone platforms.
- **KMS:** Kurento Media Server is based on pluggable media processing capabilities meaning that any of its provided features is a pluggable module that can be activated or deactivated.
- **OpenCV:** OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage and is now maintained by Itseez. The library is cross-platform and free for use under the open-source BSD license.
- **OpenFace:** OpenFace is an open source facial behavior analysis toolkit. OpenFace is the first toolkit capable of facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation with available source code.

- **Git:** Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development.
- **JSON:** JavaScript Object Notation is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication, including as a replacement for XML in some AJAX-style systems.
- **DataURI:** The data URI scheme is a uniform resource identifier (URI) scheme that provides a way to include data in-line in web pages as if they were external resources. It is a form of file literal.
- **GUI:** Graphical User Interface
- **UML:** Unified Modelling Language
- **SQL:** Structured Query Language

# 2. Packages

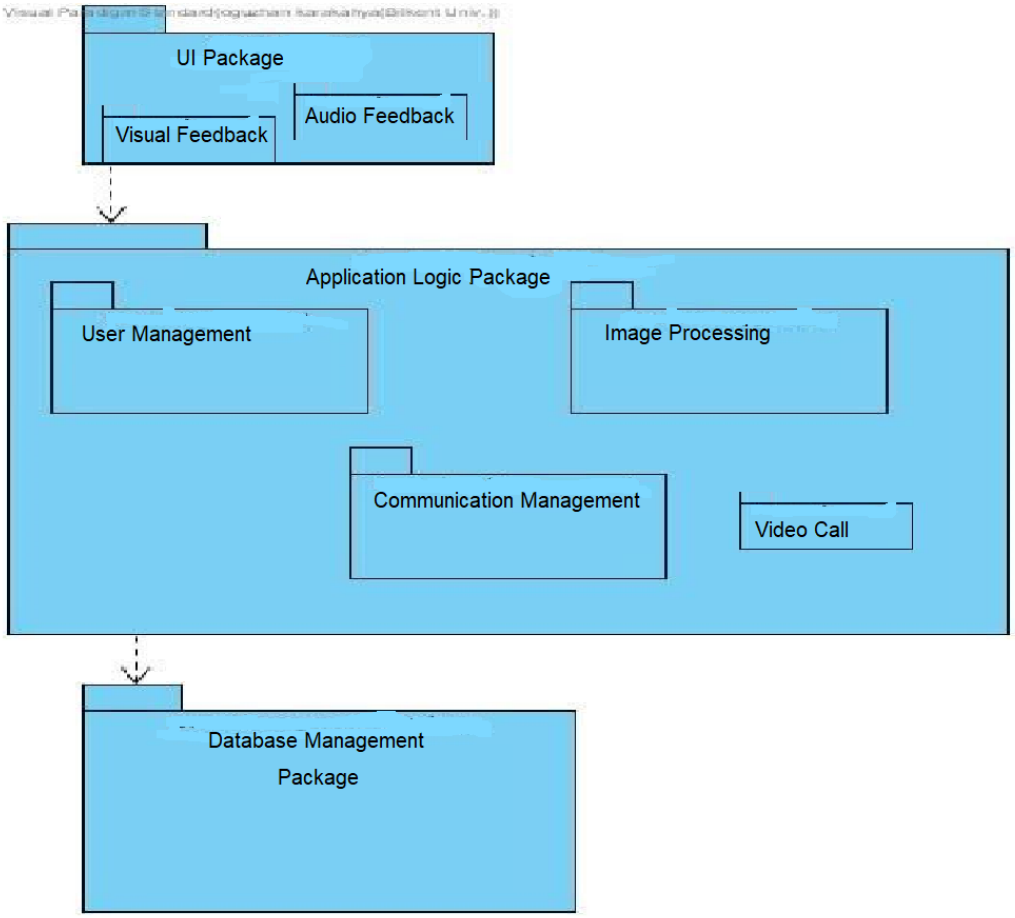


Figure 1: Packages and Dependencies Overview

In subsystem decomposition, the project is decomposed into three components. On the client side, presentation layer and application logic layer reside. Data storage layer lies under the server side.

Using this decomposition, the project is divided into three main packages as well: UI package, application logic package and data management package. Application logic package consists of three packages namely user management package, image processing package communication management package and video call package.

## 2.1. UI Packages

UI package contains classes for two different interfaces. The first one is desktop application UI. It contains classes for different application options such as audio feedback and visual feedback and its general user interface.

### 2.1.1 Visual Feedback Packages

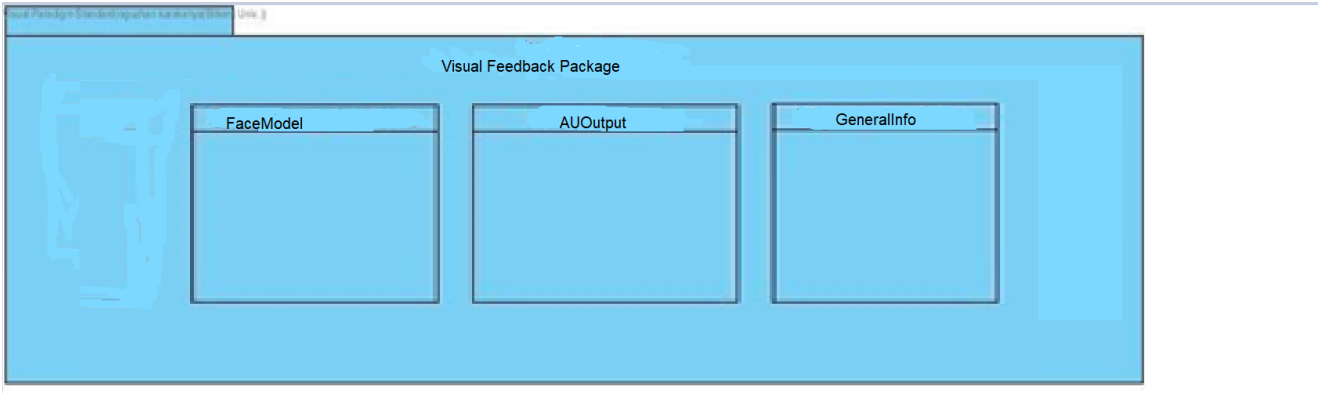


Figure 2: Visual Feedback Package

Visual Feedback Package is basically designed to give visual output to the user using FaceModel, AuOutput and GeneralInfo classes. It can be further seen down below what class is qualified on which output.

FaceModel: FaceModel is designed to create an animation face showing the facial attributes such as glasses/none, beard, hair color, eye color, age, gender, skin color, etc.

AuOutput: This is an output system coming with the OpenFace itself. OpenFace describes the emotions with decimal numbers starting with Au and returns them as output.

GeneralInfo: General info is the base of the FaceModel. It is the main class which gives the facial attribute outputs to the FaceModel's animation face.

### 2.1.2. Audio Feedback Packages

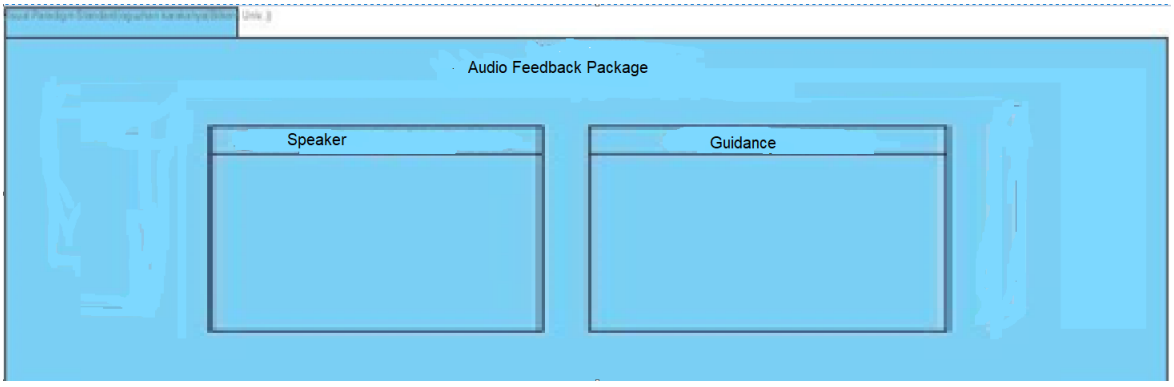


Figure 3: Audio Feedback Package

Audio Feedback Package is basically designed to give audio output to the user using Speaker and Guidance classes. It can be further seen down below what class is qualified on which output.

Speaker: Speaker class is designed to give audio output by transferring the text format to speech.

Guidance: Guidance is all about detecting if the person on the other line, who the visually impaired is speaking, is looking directly to him or not.

### 2.1.3. General User Interface Packages

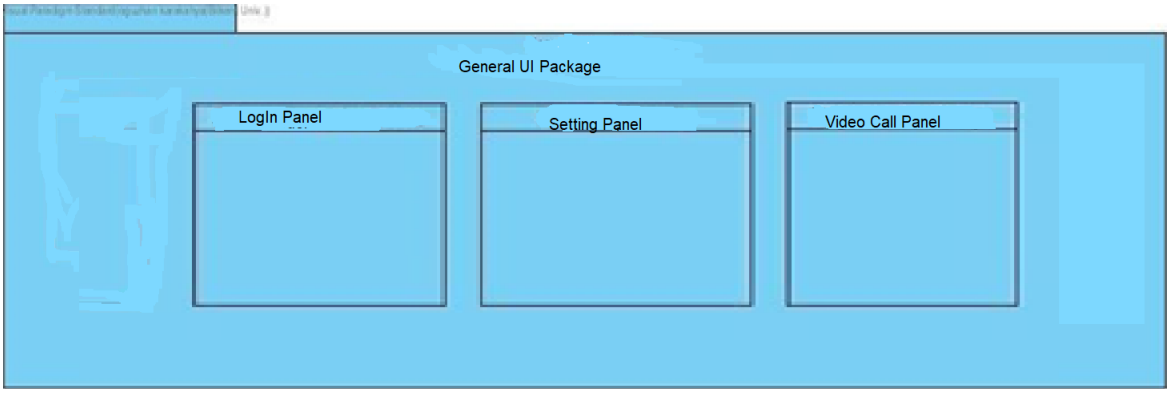


Figure 4: General UI Package

General User Interface package handles the login process, settings and the base of the project, video call.

Login Panel: This panel is responsible for the google authentication.

Setting Panel: It is responsible for the feature toggling.

VideoCall Panel : Consists a video call button to make the call start, the video will be seen from here and there will also be images.

## 2.2. Application Logic Package

This package contains the general logic files such as user management system, image processing package and communication management package and video call package. This package depends on the data management package since required user information are stored within the remote server. Data storage package methods invokes required methods to retrieve information from database and hand it over to application layer.

### 2.2.1 User Management Package

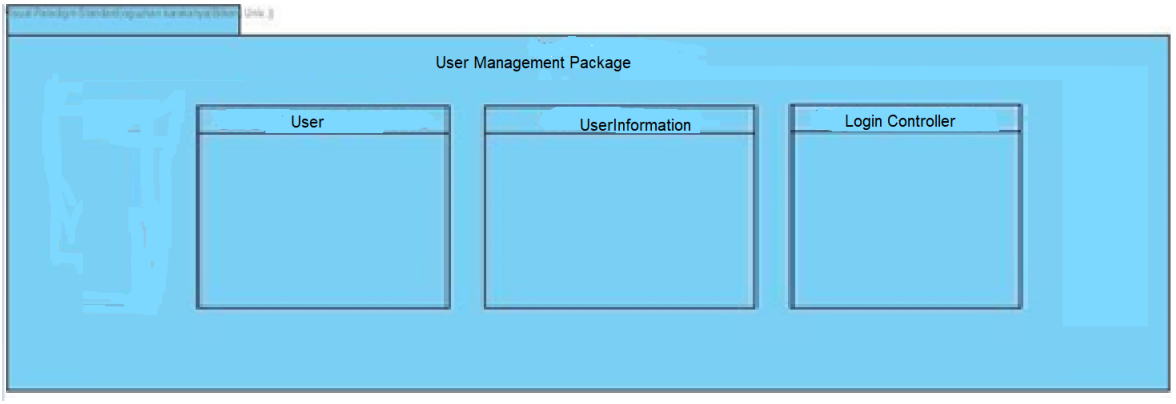


Figure 5: User Management Package and Content

This package handles user management. User is the object of the server. User Information is the object of the client. Login controller controls the login activities.

### 2.2.2. Image Processing Package

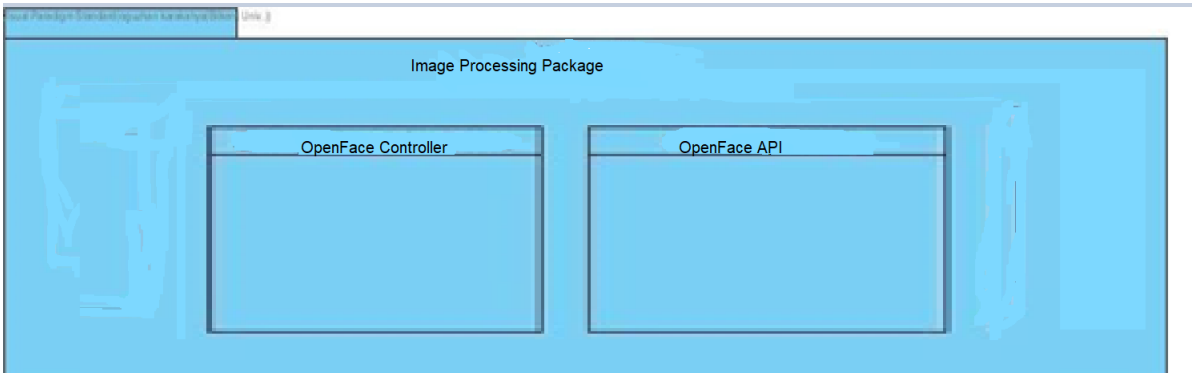


Figure 6: Image Processing Package and Content

This is one of the main packages that the project is built on. OpenFace API is used for face detection within the placed facial landmarks on the videos. OpenFace gives the visual feedback such as face detection, emotion detection. Therefore it needs to have a controller within. OpenFace Controller is an object, which OpenFace is called from, which makes sure that the OpenFace works efficiently in nearly real time.

### 2.2.3. Communication Management Package

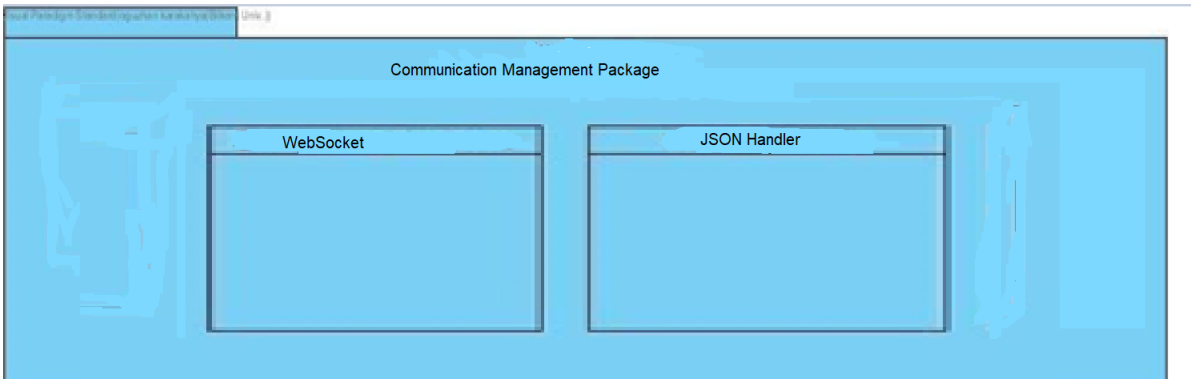


Figure 7: Communication Management Package and Content

JSON Handler: This class handles the JSON messages being generated and parsed.

WebSocket: This is an object that is responsible for the JSON message transfer.

### 2.2.4 Video Call Package

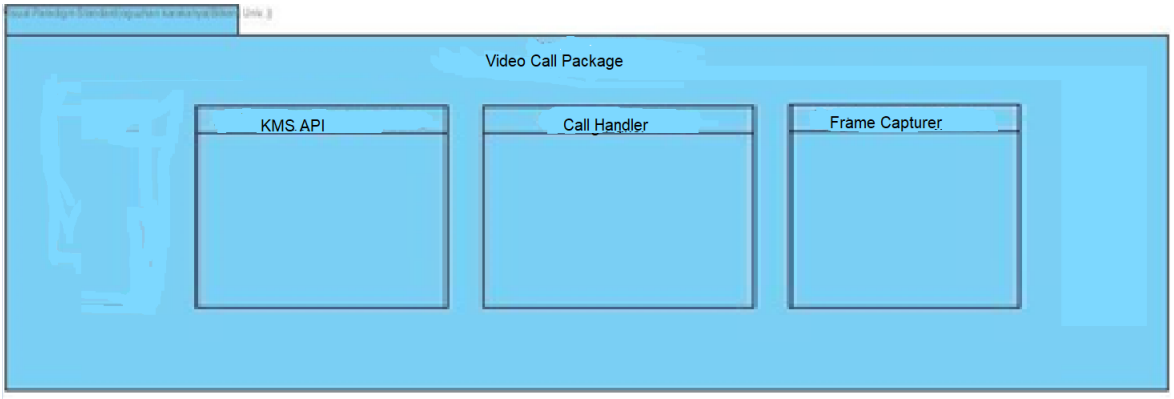


Figure 8: Video Call Package

This package is one other of the main important packages of the project. This packages is used to create the basis of the project video calling on desktop.

KMS API: This API object is responsible for signaling in the video call.

CallHandler: This object is responsible for starting or ending the call.

FrameCapturer: This object is responsible for capturing the frames.

## 2.3. Database Management Package

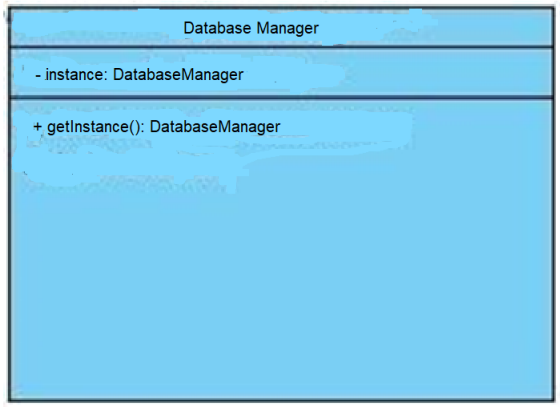


Figure 9: Database Management Package

User data is stored in a database. This package contains classes to handle database information retrieval, insertion and update on user information, frames, other contents and contacts. This package is the lowest level package, therefore it does not depend on any other
package. Only requirement is an internet connection to retrieve data from remote server.

# 3. Class Interfaces

| | |
|---|---|
| **Package** | Image Processing Package |
| **Class Name** | OpenFaceController |
| **Description** | It is the core ImageProcessing class to process frames of video call. |

| | |
|---|---|
| **Attributes** | **-** Sring frameData<br>**-** String outputData<br>**-** OpenFace API Instance |
| **Methods** | + parseFrameData(String frameData): File<br>+ generateOutptData(): void<br>+ toAPI(): boolean<br>+ fromAPI(): boolean |

Table 1: OpenFace Controller Class Interface

| | |
|---|---|
| **Package** | Image Processing Controller |
| **Class Name** | OpenFace API |
| **Description** | It is the OpenFace API connecting to OpenFace class |
| **Attributes** | **-** File frame<br>**-** File output |

| | |
|---|---|
| **Methods** | # processFrame(File frame): File |

Table 2: OpenFaceAPI Class Interface

| | |
|---|---|
| **Package** | Communication Management |
| **Class Name** | WebSocket |
| **Description** | Class that provides JSON communication. |
| **Attributes** | **- message**: String<br>**- WebSocket API instance** |
| **Methods** | # sendMessage(String message)<br># onMessage(String message): Event |

Table 3: WebSocket Class Interface

| | |
|---|---|
| **Package** | Communication Management |
| **Class Name** | JSONHandler |
| **Description** | Wrapper class to manipulate JSON objects |
| **Attributes** | **-** JSON API instance<br>**-** JSON jsonObject |
| **Methods** | # stringify(JSON jsonObject): String<br># parse(String message): JSON |

Table 4: JSONHandler Class Interface

| | |
|---|---|
| **Package** | VideoCall |
| **Class Name** | KMSAPI |
| **Description** | API class to connect to KMS. |
| **Attributes** | **- KMS API instance**<br>**- kmsURL:** String |
| **Methods** | **+ connect(String kmsURL):** boolean<br>**+ disconnect(String kmsURL):** boolean |

Table 5: KMSAPI Class Interface

| | |
|---|---|
| **Package** | VideoCall |
| **Class Name** | CallHandler |
| **Description** | Class to handle video call ending, starting etc. |
| **Attributes** | **- User** user<br>**- UserInfo** peer<br>**- KMSAPI** instance<br>**- FrameCapturer** instance |
| **Methods** | **+ startCall(User user, UserInfo peer):** JSON<br>**+ stopCall(User user, UserInfo peer):** JSON |

Table 6: CallHandler Class Interface

| | |
|---|---|
| **Package** | VideoCall |
| **Class Name** | FrameCapturer |
| **Description** | Class to capture frames during video call. |
| **Attributes** | **- frameData: String** |
| **Methods** | **+ captureFrame():** JSON<br>**+ frameToDataURI(JSON frame):** String |

Table 7: FrameCapturer Class Interface

# 4. Design Patterns

## 4.1.Façade Design Pattern

Through the Façade design pattern, a unified interface is provided to the other subsystems such as User Management Subsystem. This way, Data Management Subsystem is provided with a simpler interface and handled by DatabaseManager class.

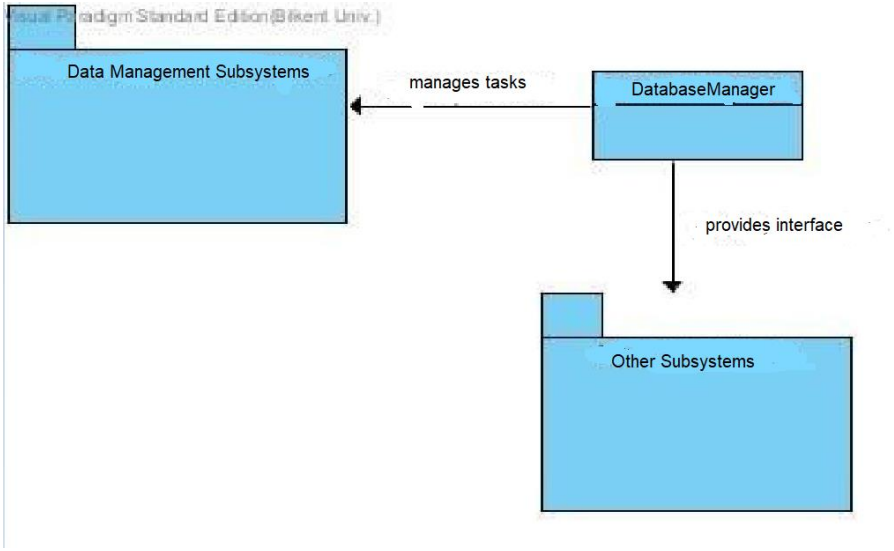


Figure 10: Façade Design Pattern

## 4.2. Singleton Design Pattern

Singleton design pattern was used in *DatabaseManager* class. Through the Singleton design pattern, it is ensured that *DatabaseManager* class has only one instance. Also, through the *getInstance()* method provides a global access to the single instance. The system needs this design pattern to ensure only one instance of *DatabaseManager* class due to the fact that the whole system will use the same manager for the database.

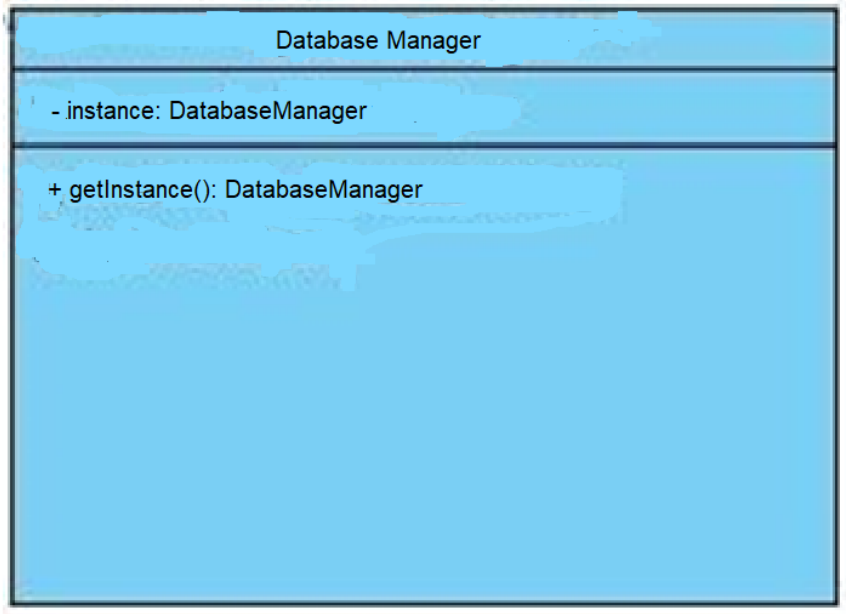


Figure 11: Singleton Design Pattern

# 5. References

[1] "CS491 Senior Design Project I", Ccs.bilkent.edu.tr, 2017. [Online]. Available: http://www.cs.bilkent.edu.tr/CS491-2/CS491.html. [2] "WebRTC", Wikipedia, 2017. [Online]. Available: http://www.wikizero.org/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2k vV2ViUlRD.