



UNIVERSITY OF CRAIOVA FACULTY OF AUTOMATICS,
COMPUTERS AND ELECTRONICS

DEPARTMENT OF COMPUTERS AND INFORMATION
TECHNOLOGY



BACHELOR`S THESIS

Dragomir Sorin Alexandru

SCIENTIFIC COORDINATOR

Prof. Univ. Dr. Ing. Brezovan Marius Vasile

July 2018

CRAIOVA



UNIVERSITY OF CRAIOVA FACULTY OF AUTOMATICS,
COMPUTERS AND ELECTRONICS

DEPARTMENT OF COMPUTERS AND INFORMATION
TECHNOLOGY



Emotions Recognition in Images

Dragomir Sorin Alexandru

SCIENTIFIC COORDINATOR

Prof. Univ. Dr. Ing. Brezovan Marius Vasile

July 2018

CRAIOVA

“Never regard study as a duty, but as an enviable opportunity to learn to know the liberating influence of beauty in the realm of the spirit for your own personal joy and to the profit of the community to which your later works belong.”

Albert Einstein

DECLARAȚIE DE ORIGINALITATE

Subsemnatul *DRAGOMIR SORIN ALEXANDRU*, student la specializarea *CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI* din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoștință de cele prezentate mai jos și că îmi asum, în acest context, originalitatea proiectului meu de licență:

- cu titlul *EMOTIONS RECOGNITION IN IMAGES*,
- coordonată de *PROF. UNIV. DR. ING. MARIUS BREZOVAN*,
- prezentată în sesiunea *IULIE 2018*.

La elaborarea proiectului de licență, se consideră plagiat una dintre următoarele acțiuni:

- reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele și referința precisă,
- redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
- prezentarea unor date experimentale obținute sau a unor aplicații realizate de alți autori fără menționarea corectă a acestor surse,
- însușirea totală sau parțială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situații neplăcute se recomandă:

- plasarea între ghilimele a citatelor directe și indicarea referinței într-o listă corespunzătoare la sfârșitul lucrării,
- indicarea în text a reformulării unei idei, opinii sau teorii și corespunzător în lista de referințe a sursei originale de la care s-a făcut preluarea,
- precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
- precizarea referințelor poate fi omisă dacă se folosesc informații sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data,

Semnătura candidatului,

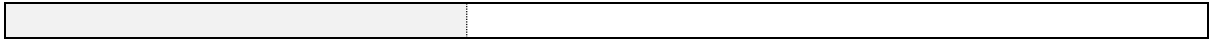


UNIVERSITATEA DIN CRAIOVA
Facultatea de Automatică, Calculatoare și Electronică
Departamentul de Calculatoare și Tehnologia Informației

Aprobat la data de
Șef de departament,
Prof. dr. ing.
Marius BREZOVAN

PROIECTUL DE DIPLOMĂ

Numele și prenumele studentului/-ei:	<i>Dragomir Sorin Alexandru</i>
Enunțul temei:	<i>Emotions Recognition in Images</i>
Datele de pornire:	<i>Datele de pornire ale aplicației au fost propunerea temei de către domnul Prof. Univ. Dr. Ing. Brezovan Marius Vasile și dorința de a învăța lucruri noi. De asemenea, tema aleasă este una antrenantă, tot mai des întâlnită atât în diverse aplicații Web sau Mobile, cât și în domeniul Roboticii.</i>
Conținutul proiectului:	<i>Enunțul aplicației – specificarea cerințelor aplicației, motivația și scopul; Concepte și unelte de dezvoltare - prezentarea cerințelor tehnice ale proiectului; Implementarea aplicației - descrierea detaliată a aplicației și a aspectelor arhitecturale; Referințe.</i>
Material grafic obligatoriu:	Diagrame UML, scheme, slide-uri, imagini
Consultații:	<i>Periodice</i>
Conducătorul științific (titlul, nume și prenume, semnătura):	Prof. Univ. Dr. Ing. Brezovan Marius Vasile
Data eliberării temei:	15.11.2017
Termenul estimat de predare a proiectului:	05.07.2018
Data predării proiectului de către student și semnătura acestuia:	





REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC

Numele și prenumele candidatului/-ei:

Dragomir Sorin Alexandru

Specializarea:

Calculatoare (în limba engleză)

Titlul proiectului:

Emotions Recognition in Images

Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta):

În facultate ☐

În producție ☐

În cercetare ☐

Altă locație:

În urma analizei lucrării candidatului au fost constatate următoarele:

Nivelul documentării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Tipul proiectului		Cercetare <input type="checkbox"/>	Proiectare <input type="checkbox"/>	Realizare practică <input type="checkbox"/>	Altul
Aparatul matematic utilizat		Simplu <input type="checkbox"/>	Mediu <input type="checkbox"/>	Complex <input type="checkbox"/>	Absent <input type="checkbox"/>
Utilitate		Contract de cercetare <input type="checkbox"/>	Cercetare internă <input type="checkbox"/>	Utilare <input type="checkbox"/>	Altul
Redactarea lucrării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Partea grafică, desene		Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
Realizarea practică	Contribuția autorului	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Mare <input type="checkbox"/>	Foarte mare <input type="checkbox"/>
	Complexitatea temei	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>
	Analiza cerințelor	Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
	Arhitectura	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>

	Întocmirea specificațiilor funcționale	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Implementarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Testarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Funcționarea	Da <input type="checkbox"/>	Parțială <input type="checkbox"/>	Nu <input type="checkbox"/>	
Rezultate experimentale		Experiment propriu <input type="checkbox"/>		Preluare din bibliografie <input type="checkbox"/>	
Bibliografie		Cărți	Reviste	Articole	Referințe web
Comentarii și observații					

În concluzie, se propune:

ADMITEREA PROIECTULUI <input type="checkbox"/>	RESPINGEREA PROIECTULUI <input type="checkbox"/>
---	---

Data,

Semnătura conducătorului științific,

SUMMARY

This application is a Web Application which recognizes the 8 main human emotions (anger, contempt, disgust, fear, happiness, neutral, sadness, surprise) in images.

The user can either upload an image from the disk or insert the url of an image from the internet in order to be analyzed. A third option for the users would be to use the webcam to take an instant capture and use the application together with their real time facial expressions.

The algorithm used for the image processing, face detection and emotion recognition is the Face API from Microsoft. This API can detect one or more human faces in an image and get back face rectangles for where in the image the faces are, along with face attributes which contain machine learning-based predictions of facial features. The face attribute features available are: Age, Emotion, Gender, Pose, Smile, and Facial Hair along with 27 landmarks for each face in the image.

The final output represents an emotions summary. From all the detected emotions one is remarkable as being predominant and it is also displayed. The user can also choose to display other face attributes besides emotions such as: Age, Gender, Facial Hair, Facial Accessories.

Key words: *web application, face detection, face attributes, emotion recognition, Face Api (Microsoft).*

ACKNOWLEDGEMENTS

Firstly, I would like to thank the teachers from the Faculty of Automatics, Computers and Electronics for their guidance and teaching during my four years of studying. I consider I have developed a lot of important programming skills and improved the way to think and tackle a problem during all the courses and laboratories that were held during the four years of study.

I would also like to thank my coordinator and advisor during this license project, Prof. Univ. Dr. Ing. Brezovan Marius Vasile, for all his help and guidance during the development of the project.

Finally, I would like to thank my fellow students for keeping me focused and motivated to learn new things even in difficult moments.

TABLE OF CONTENTS

1	INTRODUCTION	16
1.1	PURPOSE	16
1.2	MOTIVATION	16
2	TOOLS, TECHNOLOGIES, INTEGRATED DEVELOPMENT ENVIRONMENTS	18
2.1	ASP.NET MVC	18
2.1.1	<i>Why ASP.NET MVC?</i>	19
2.1.2	<i>Visual Studio</i>	20
2.2	C#	21
2.2.1	<i>C# features</i>	23
2.3	ABOUT APIS	26
2.3.1	<i>Face API Microsoft features</i>	32
2.3.2	<i>Used features</i>	32
2.4	BOOTSTRAP	33
2.4.1	<i>Why Bootstrap?</i>	36
2.5	HTML	38
2.6	CSS	39
2.7	JAVASCRIPT	41
2.7.1	<i>jQuery</i>	47
2.8	VERSION CONTROL	49
2.8.1	<i>GIT</i>	50
3	REQUIREMENTS AND APPLICATION SCHEME	51
3.1	REQUIREMENTS	51
3.2	FUNCTIONAL REQUIREMENTS	51
3.3	APPLICATION FLOW	53
4	APPLICATION DEVELOPMENT	54
4.1	CHOOSING FACE API MICROSOFT	54
4.2	FRONT END	55
4.2.1	<i>Change UI theme</i>	55
4.2.2	<i>Upload image</i>	58
4.2.3	<i>Webcam capture</i>	65
4.2.4	<i>Display results</i>	70
4.3	BACK END	74
4.3.1	<i>Analyze Image</i>	74

5	LICENSE	85
6	BIBLIOGRAPHY	86
7	WEB REFERENCES	87
8	CD / DVD	89
9	INDEX	90

Table of figures

MODEL VIEW CONTROLLER 2-1	18
MODEL VIEW CONTROLLER 2-2	19
VISUAL STUDIO 2-1	20
C# 2-1	21
API 2-1	26
BOOTSTRAP 2-1	33
DOM 2-1	38
HTML 2-1	38
CSS 2-1	39
CSS 2-2	40
JAVASCRIPT 2-1	41
JAVASCRIPT LIMITATIONS 2-1	45
JQUERY 2-1	47
VERSION CONTROL 2-1	49
GIT 2-1	50
GITHUB 2-1	50
APPLICATION FLOW 3-1	53
BOOTSTRAP THEME 4-1	56
BOOTSTRAP THEME 4-2	57
AJAX CALL 4-1	58
AJAX CALL 4-2	63
CONTROLLER ACTION 4-1	63
CONTROLLER ACTION 4-2	64
SHOW LOADER METHOD 4-1	62
HIDE LOADER 4-1	62
GET MORE INFO 4-1	72
OPEN WEBCAM 4-1	65
TAKE WEBCAM SNAPSHOT 4-1	66
CLOSE WEBCAM 4-1	67
WEBCAM ERROR 4-1	68
WEBCAM 4-1	69
ANALYSIS RESULTS 4-1	70
ANALYSIS RESULTS 4-2	70
ANALYSIS RESULTS 4-4	71
ANALYSIS RESULTS 4-5	71

GET MORE INFO 4-1.....	72
FACE RECTANGLES 4-1.....	73

1 INTRODUCTION

1.1 Purpose

This paper work represents my bachelor's degree project documentation after studying four years at the Faculty of Automatics, Computers and Electronics, University of Craiova. Its purpose is to provide all necessary information for the readers regarding the architecture, technologies, design and functionality of the project.

Furthermore, I would also like to explain the reason I choose this project, which could be his use in real life and how it could help its users to optimize its activities.

Finally, this project will test my abilities as a developer, designer, project manager and even tester in the same time and I will need to find innovative solutions and quickly learn new technologies and platforms.

1.2 Motivation

The theme of the application was proposed by Prof. Univ. Dr. Ing. Brezovan Marius Vasile and together with my desire of learning new things I decided to develop a Web Application whose main purpose is Emotions Recognition in Images. Also, the chosen theme is an exciting one, more and more common in both Web and Mobile applications, as well as in the Robotics domain.

In the last years I have noticed the increased interest in emotion detect systems which can analyze basic facial expression of human together with face recognition function. These systems can be used in Web or Mobile applications such as: emotion-based recommendations applications, social media applications etc. However, the highest applicability, in my opinion, is in the Robotics domain. These systems enables the robots not only to recognize human emotions, but also to generate facial expression for adapting to human emotions.

Emotion recognition takes mere facial detection/recognition a step further, and its use cases are nearly endless.

An obvious use case is within group testing. User response to video games, commercials, or products can all be tested at a larger scale, with large data accumulated automatically, and thus more efficiently. Bentley used facial expression recognition in a marketing campaign to suggest car model types based on emotive responses to certain stimuli. Technology that reveals your feelings has also been suggested to spot struggling students in a classroom environment, or help autistics better interact with others. Some use cases include:

- Helping to better measure TV ratings.
- Adding another security layer to security at malls, airports, sports arenas, and other public venues to detect malicious intent.
- Wearables that help autistics discern emotion
- Check out counters, virtual shopping
- Creating new virtual reality experiences

Those listed above together with the evolution trend of the AI and machine learning made me think that it would be an interesting application for my diploma project.

2 TOOLS, TECHNOLOGIES, INTEGRATED DEVELOPMENT ENVIRONMENTS

2.1 ASP.NET MVC

ASP stands for Active Server Pages and it is a development framework for building web pages.

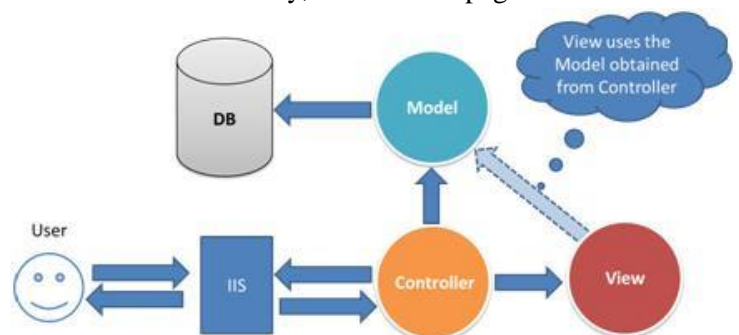
ASP supports many different development models:

- Classic ASP
- ASP.NET Web Forms
- ASP.NET MVC
- ASP.NET Web Pages
- ASP.NET API
- ASP.NET Core

“ASP and ASP.NET are server side technologies. Both technologies enable computer code to be executed by an Internet server. When a browser requests an ASP or ASP.NET file, the ASP engine reads the file, executes any code in the file, and returns the result to the browser.”

The ASP.NET MVC is a web application framework developed by Microsoft, which implements the model–view–controller (MVC) pattern. It is open-source software, apart from the ASP.NET Web Forms component which is proprietary.

Razor is a markup syntax that lets you embed server-based code (Visual Basic and C#) into web pages. Server-based code can create dynamic web content on the fly, while a web page is written to the browser. When a web page is called, the server executes the server-based code inside the page before it returns the page to the browser. By running on the server, the code can perform complex tasks, like accessing databases. Razor is based on ASP.NET, and designed for creating web applications.



Model View Controller 2-1

The MVC Programming Model

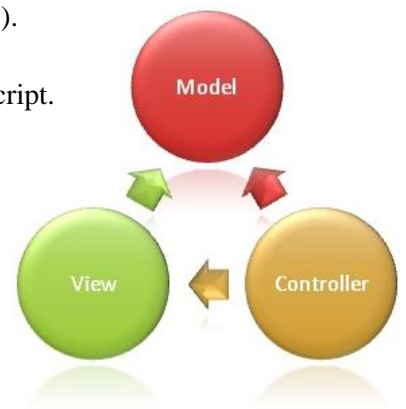
MVC is a framework for building web applications using a MVC (Model View Controller) design:

- The Model represents the application core (for instance a list of database records).
- The View displays the data (the database records).
- The Controller handles the input (to the database records).

The MVC model also provides full control over HTML, CSS, and JavaScript.

The MVC model defines web applications with 3 logic layers:

- The business layer (Model logic)
- The display layer (View logic)
- The input control (Controller logic)



Model View Controller 2-2

The Model is the part of the application that handles the logic for the application data. Often model objects retrieve data (and store data) from a database.

The View is the parts of the application that handles the display of the data. Most often the views are created from the model data.

The Controller is the part of the application that handles user interaction. Typically controllers read data from a view, control user input, and send input data to the model.

2.1.1 Why ASP.NET MVC?

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and that gives you full control over markup for

enjoyable, agile development. ASP.NET MVC includes many features that enable fast, TDD-friendly development for creating sophisticated applications that use the latest web standards.

“Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only. This is opposed to software development that allows software to be added that is not proven to meet requirements.”

The MVC separation helps you manage complex applications, because you can focus on one aspect a time. For example, you can focus on the view without depending on the business logic. It also makes it easier to test an application.

The MVC separation also simplifies group development. Different developers can work on the view, the controller logic, and the business logic in parallel.

2.1.2 Visual Studio



Visual Studio 2-1

“Visual Studio is an IDE, or integrated development environment. Just like Microsoft Word is used to write documents, an IDE is used to create applications.”

Visual Studio is used for Windows and Mac and it offers support to develop applications for Android, iOS, Mac, Windows, web, and cloud.

See the App Running on Azure

In order to see the finished site running as a live web app you can deploy a complete version of the app to an Azure account.

“Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. It provides software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.”

Azure was announced in October 2008 and released in February 1, 2010 as "Windows Azure" before being renamed "Microsoft Azure" on March 25, 2014.

2.2 C#



C# 2-1

“C# is a modern, general-purpose, object-oriented programming language developed by Microsoft and approved by European Computer Manufacturers Association (ECMA) and International Standards Organization (ISO).”

C# was developed by Anders Hejlsberg and his team during the development of .Net Framework.

C# is designed for Common Language Infrastructure (CLI), which consists of the executable code and runtime environment that allows use of various high-level languages on different computer platforms and architectures.

The following reasons make C# a widely used professional language:

- It is a modern, general-purpose programming language
- It is object oriented.
- It is component oriented.
- It is easy to learn.
- It is a structured language.
- It produces efficient programs.
- It can be compiled on a variety of computer platforms.
- It is a part of .Net Framework.

Strong Programming Features of C#

Although C# constructs closely follow traditional high-level languages, C and C++ and being an object-oriented programming language. It has strong resemblance with Java, it has numerous strong programming features that make it endearing to a number of programmers worldwide.

Following is the list of few important features of C#:

- Boolean Conditions
- Automatic Garbage Collection
- Standard Library
- Assembly Versioning
- Properties and Events
- Delegates and Events Management
- Easy-to-use Generics
- Indexers
- Conditional Compilation
- Simple Multithreading
- LINQ and Lambda Expressions
- Integration with Windows

The .Net Framework

The .Net framework is a revolutionary platform that helps you to write the following types of applications:

- Windows applications
- Web applications
- Web services

The .Net framework applications are multi-platform applications. The framework has been designed in such a way that it can be used from any of the following languages: C#, C++, Visual

Basic, Jscript, COBOL, etc. All these languages can access the framework as well as communicate with each other.

The .Net framework consists of an enormous library of codes used by the client languages such as C#. Following are some of the components of the .Net framework:

- Common Language Runtime (CLR)
- The .Net Framework Class Library
- Common Language Specification
- Common Type System
- Metadata and Assemblies
- Windows Forms
- ASP.Net and ASP.Net AJAX
- ADO.Net
- Windows Workflow Foundation (WF)
- Windows Presentation Foundation
- Windows Communication Foundation (WCF)
- LINQ

Writing C# Programs on Linux or Mac OS

Although the .NET Framework runs on the Windows operating system, there are some alternative versions that work on other operating systems. Mono is an open-source version of the .NET Framework which includes a C# compiler and runs on several operating systems, including various flavors of Linux and Mac OS. Kindly check Go Mono.

The stated purpose of Mono is not only to be able to run Microsoft .NET applications cross-platform, but also to bring better development tools for Linux developers. Mono can be run on many operating systems including Android, BSD, iOS, Linux, OS X, Windows, Solaris, and UNIX.

2.2.1 C# features

- C# is a simple, modern, object oriented language derived from C++ and Java.

- It aims to combine the high productivity of Visual Basic and the raw power of C++.
- It is a part of Microsoft Visual Studio 7.0.
- Visual Studio supports VB, VC++, C++, VBscript, JScript. All of these languages provide access to the Microsoft .NET platform.
- .NET includes a Common Execution engine and a rich class library.
- Microsofts JVM equiv is Common language run time(CLR).
- CLR accommodates more than one languages such as C#, VB.NET, Jscript, ASP.NET, C++.
- Source code --->Intermediate Language code(IL) ---> (JIT Compiler) Native code.
- The classes and data types are common to all of the .NET languages.
- We may develop Console application, Windows application, and Web application using C#.
- In C#, Microsoft has taken care of C++ problems, such as Memory management, pointers etc.
- It supports garbage collection, automatic memory management, and a lot.

Main features of C#

1. Simple

- Pointers are missing in C#.
- Unsafe operations such as direct memory manipulation are not allowed.
- In C# there is no usage of "::" or "->" operators.
- Since it's on .NET, it inherits the features of automatic memory management and garbage collection.
- Varying ranges of the primitive types like Integer, Floats etc.
- Integer values of 0 and 1 are no longer accepted as boolean values. Boolean values are pure true or false values in C# so no more errors of "="operator and "=="operator.

- "==" is used for comparison operation and "=" is used for assignment operation.

2. Modern

- C# has been based according to the current trend and is very powerful and simple for building interoperable, scalable, robust applications.
- C# includes built in support to turn any component into a web service that can be invoked over the internet from any application running on any platform.

3. Object oriented

- C# supports Data Encapsulation, inheritance, polymorphism, interfaces.
- (int, float, double) are not objects in java but C# has introduced structures (structs) which enable the primitive types to become objects.

```
int i=1;
```

```
string a=i.ToString(); //conversion (or) Boxing
```

4. Type safe

- In C# we cannot perform unsafe casts like convert double to a boolean.
- Value types (primitive types) are initialized to zeros and reference types (objects and classes) are initialized to null by the compiler automatically.
- Arrays are zero base indexed and are bound checked.
- Overflow of types can be checked.

5. Interoperability

- C# includes native support for the COM and windows based applications.
- Allowing restricted use of native pointers.
- Users no longer have to explicitly implement the unknown and other COM interfaces, those features are built in.

- C# allows the users to use pointers as unsafe code blocks to manipulate your old code.
- Components from VB NET and other managed code languages and directly be used in C#.

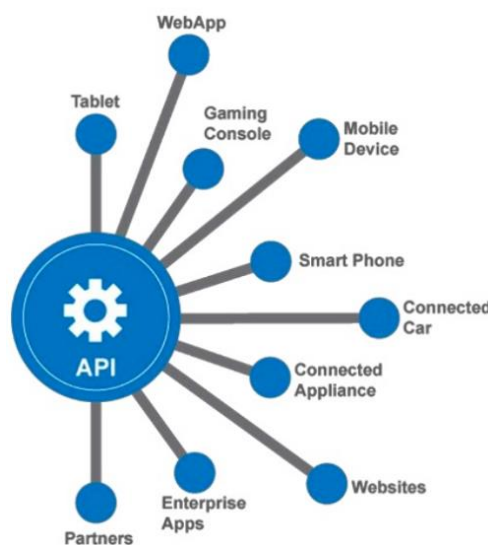
6. Scalable and updateable

- .NET has introduced assemblies which are self describing by means of their manifest. manifest establishes the assembly identity, version, culture and digital signature etc. Assemblies need not to be register anywhere.
- To scale our application we delete the old files and updating them with new ones. No registering of dynamic linking library.
- Updating software components is an error prone task. Revisions made to the code. can effect the existing program C# support versioning in the language. Native support for interfaces and method overriding enable complex frame works to be developed and evolved over time.

Conclusion

C# is a modern, type safe programming language, object oriented language that enables programmers to quickly and easily build solutions for the Microsoft .NET platform.

2.3 About APIs



API 2-1

“API stands for Application Programming Interface, which specifies how one component should interact with the other. It consists of a set of routines, protocols and tools for building the software applications.”

Put briefly, an API consists of a set of rules describing how one application can interact with another, and the mechanisms that allow such interaction to happen.

What is an interaction between two applications? Typically, an interaction occurs when one application would like to access the data held by another application, or send data to that app. Another interaction might be when one application wants to request a service from another.

A key thing to note: An API is (usually) not a user interface. It provides software-to-software interaction, not user interactions. Sometimes, though, an API may provide a user interface widget, which an app can grab and display.

There are two primary benefits that an API brings:

- Simplification, by providing a layer that hides complexity.
- Standardisation.

Examples:

- Microsoft Word asks the active printer to return its status. Microsoft Word does not care what kind of printer is available. The API worries about that.
- Bloggers on WordPress can embed their Twitter stream into their blog’s sidebar. WordPress uses the Twitter API to enable this.

Web service APIs

“A web service is a piece of software, or a system, that provides access to its services via an address on the World Wide Web. This address is known as a URI, or URL. The key point is that the web service offers its information in a format that other applications can “understand”, or parse.”

Examples: The Flickr API, the Google Static Maps API, and the other Google Maps web services.

A web service uses HTTP to exchange information. (Or HTTPS, which is an encrypted version of HTTP.)

When an application, the “client”, wants to communicate with the web service, the application sends an HTTP request. The web service then sends an HTTP response.

In the request, much of the required information is passed in the URL itself, as paths in the URL and/or as URL parameters.

For example:

`http://maps.googleapis.com/maps/api/staticmap?center=Sydney,NSW&zoom=14&size=400x400&sensor=false`

In addition to the URL, HTTP requests and responses will include information in the header and the body of the message. Request and response “headers” include various types of metadata, such as the browser being used, the content type, language (human, not software), and more.

The body includes additional data in the request or response. Common data formats are XML and JSON. The process of converting data from internal format (for example, a database or a class) to the transferrable format is called “data serialization”.

Most often-used types of web service:

- SOAP
- XML-RPC
- JSON-RPC
- REST

SOAP (Simple Object Access Protocol)

“SOAP is a protocol that defines the communication method, and the structure of the messages. The data transfer format is XML.”

A SOAP service publishes a definition of its interface in a machine-readable document, using WSDL – Web Services Definition Language.

XML-RPC

“XML-RPC is an older protocol than SOAP. It uses a specific XML format for data transfer, whereas SOAP allows a proprietary XML format. An XML-RPC call tends to be much simpler, and to use less bandwidth, than a SOAP call. (SOAP is known to be “verbose”.) SOAP and XML-RPC

have different levels of support in various libraries. There's good information in this Stack Overflow thread."

JSON-RPC

JSON-RPC is similar to XML-RPC, but uses JSON instead of XML for data transfer.

REST (Representational state transfer)

"REST is not a protocol, but rather a set of architectural principles. The thing that differentiates a REST service from other web services is its architecture. Some of the characteristics required of a REST service include simplicity of interfaces, identification of resources within the request, and the ability to manipulate the resources via the interface. There are a number of other, more fundamental architectural requirements too."

Looked at from the point of view of a client application, REST services tend to offer an easy-to-parse URL structure, consisting primarily of nouns that reflect the logical, hierarchical categories of the data on offer.

For example, let's say you need to get a list of trees from an API at example-tree-service.com. You might submit a request like this:

`http://example-tree-service.com/trees`

Perhaps you already know the scientific name of a tree family, Leptospermum, and you need to know the common name. Your request might look like this:

`http://example-tree-service.com/trees/leptospermum`

The tree service might then send a response containing a bunch of information about the Leptospermum family, including a field "common-name" containing the value "teatrees".

An example of a REST API: The JIRA REST APIs from Atlassian.

The most commonly-used data format is JSON or XML. Often the service will offer a choice, and the client can request one or the other by including "json" or "xml" in the URL path or in a URL parameter.

A REST service may publish a WADL document describing the resources it has available, and the methods it will accept to access those resources. WADL stands for Web Application Description Language. It's an XML format that provides a machine-processable description of an HTTP-based Web applications. If there's no WADL document available, developers rely on

documentation to tell them what resources and methods are available. Most web services still rely on documentation rather than a machine-readable description of their interface.

In a well-defined REST service, there is no tight coupling between the REST interface and the underlying architecture of the service. This is often cited as the main advantage of REST over RPC (Remote Procedure Call) architectures. Clients calling the service are not dependent on the underlying method names or data structures of the service. Instead, the REST interfaces merely represent the logical resources and functionality available. The structure of the data in the message is independent of the service's data structure. The message contains a representation of the data. Changes to the underlying service must not break the clients.

Other types of APIs:

Library-based APIs

To use this type of API, an application will reference or import a library of code or of binary functions, and use the functions/routines from that library to perform actions and exchange information.

JavaScript APIs are a good example. Take a look at the Google Maps JavaScript API. To display an interactive Google Map on a web page, you add a <script> tag to include the JavaScript library provided by Google. Then you write your own JavaScript code, calling the Google Maps functions as needed.

Another example is the JavaScript Datastore API from Dropbox. And the Twilio APIs offer libraries for a range of languages and frameworks, including PHP, Python, JavaScript, and many more.

TWAIN is an API and communications protocol for scanners and cameras. For example, when you buy an HP scanner you will also get a TWAIN software library, written to comply with the TWAIN standard which supports multiple device types. Applications will use TWAIN to talk to your scanner.

The Oracle Call Interface (OCI) consists of a set of C-language software APIs which provide an interface to the Oracle database.

Class-based APIs (object oriented) – a special type of library-based API

These APIs provide data and functionality organised around classes, as defined in object-oriented languages. Each class offers a discrete set of information and associated behaviours, often corresponding to a human understanding of a concept.

The Java programming community offers a number of good examples of object oriented, or classed-based, APIs. For example:

- The Java API itself. This is a set of classes that come along with the Java development environment (JDK) and which are indispensable if you're going to program in Java. The Java language includes the basic syntax and primitive types. The classes in the Java API provide everything else – things like strings, arrays, the renowned Object, and much much more.
- The Android API.
- The Google Maps Android API.

As an example for C#, there's the MSDN Class Library for the .NET Framework. The Twilio APIs mentioned above also include both Java and C#.

Functions or routines in an OS

Operating systems, like Windows and UNIX, provide many functions and routines that we use every day without thinking about it. These OSes offer an API too, so that software programs can interact with the OS.

Examples of functionality provided by the API: Access to the file system, printing documents, displaying the content of a file on the console, error notifications, access to the user interface provided by the OS.

Object remoting APIs

These APIs use a remoting protocol, such as CORBA – Common Object Request Broker Architecture. Such an API works by implementing local proxy objects to represent the remote objects, and interacting with the local object. The same interaction is then duplicated on the remote object, via the protocol. Another example is .NET Remoting.

Hardware APIs

Hardware APIs are for manipulating addressable pieces of hardware on a device – things like video acceleration, hard disk drives, PCI buses.

2.3.1 Face API Microsoft features

What is Face API?

“The Microsoft Face API is a cloud-based service that provides the most advanced face algorithms. Face API has two main functions: face detection with attributes and face recognition.”

The Microsoft Emotion API used for human emotions recognition in images was deprecated on October 30, 2017. The functionality is now part of Face API.

Before using the Face API, you must sign up to subscribe to Face API in the Microsoft Cognitive Services portal and get your subscription key.

Face detection

Detect one or more human faces in an image and get back face rectangles for where in the image the faces are, along with face attributes which contain **machine learning-based predictions of facial features**. The face attribute features available are: Age, Emotion, Gender, Pose, Smile, and Facial Hair along with 27 landmarks for each face in the image.

Emotion recognition

The Face API now integrates emotion recognition, returning the confidence across a set of emotions for each face in the image such as anger, contempt, disgust, fear, happiness, neutral, sadness, and surprise. These emotions are understood to be cross-culturally and universally communicated with particular facial expressions.

2.3.2 Used features

Emotion Recognition

“The Microsoft Face API takes an image as an input and returns the confidence across a set of emotions for each face in the image, as well as a bounding box for the face from the Face API. The emotions detected are happiness, sadness, surprise, anger, fear, contempt, disgust, or neutral. These emotions are communicated cross-culturally and universally via the same basic facial expressions, where are identified by Emotion API.”

Interpreting Results:

- In interpreting results from the Emotion API, the emotion detected should be interpreted as the emotion with the highest score, as scores are normalized to sum to one. Users may choose to set a higher confidence threshold within their application, depending on their needs.
- The Face API recognizes the emotions expressed by one or more people in an image, as well as returns a bounding box for the face. The emotions detected are happiness, sadness, surprise, anger, fear, contempt, and disgust or neutral.
- The supported input image formats includes JPEG, PNG, GIF(the first frame), BMP. Image file size should be no larger than 4MB.
- If a user has already called the Face API, they can submit the face rectangles as an optional input. Otherwise, Emotion API will first compute the rectangles.
- The detectable face size range is 36x36 to 4096x4096 pixels. Faces out of this range will not be detected.
- For each image, the maximum number of faces detected is 64 and the faces are ranked by face rectangle size in descending order. If no face is detected, an empty array will be returned.
- Some faces may not be detected due to technical challenges, e.g. very large face angles (head-pose), large occlusion. Frontal and near-frontal faces have the best results.
- The emotions contempt and disgust are experimental.

2.4 Bootstrap



Bootstrap 2-1

Bootstrap is a free and open-source front-end framework for designing websites and web applications.

“Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.”

Bootstrap represents the world’s most popular framework for building responsive, mobile-first sites, with BootstrapCDN and a template starter page.

“Bootstrap is a free and open-source front-end framework (library) for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only. Bootstrap is the second most-starred project on GitHub, with more than 123,000 stars.”

Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools.

Since 2.0, Bootstrap supports responsive web design. This means the layout of web pages adjusts dynamically, taking into account the characteristics of the device used (desktop, tablet, mobile phone).

Bootstrap is modular and consists of a series of Less stylesheets that implement the various components of the toolkit. These stylesheets are generally compiled into a **bundle** and included in web pages, but individual components can be included or removed. Bootstrap provides a number of configuration variables that control things such as color and padding of various components.

“A Web Essentials bundle file is a recipe for grouping, and usually compressing, a set of files of the same type to limit the number and the amount of the data to be downloaded by the browser.”

Web Essentials offers two bundling types:

- **.bundle** : for CSS and JS files: For CSS, it outputs a XML bundle recipe file, a destination CSS file and a minified version of source if you turn on the minify option on the recipe. For JavaScript files, it outputs a destination JS file, a minified version of sources and a source-map of that min.
- **.sprite** : for images (PNG, JPG and GIF). It generates a sprite image, CSS with example code for all the possible coordinates in background property, LESS and SASS

files with mixins holding the same background properties and a custom map file (JSON) with all those coordinates.

As of Bootstrap 4, Sass is used instead of Less for the stylesheets.

Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code.

Stylesheets

Bootstrap provides a set of stylesheets that provide basic style definitions for all key HTML components. These provide a uniform, modern appearance for formatting text, tables and form elements.

Re-usable components

In addition to the regular HTML elements, Bootstrap contains other commonly used interface elements. The components are implemented as CSS classes, which must be applied to certain HTML elements in a page.

JavaScript components

Bootstrap comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields. In version 1.3, the following JavaScript plugins are supported: Modal, Dropdown, Scrollspy, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel and Typeahead.

Bootstrap 4

On October 29, 2014, Mark Otto announced that Bootstrap 4 was in development.[8] On September 6, 2016, Mark suspended work on Bootstrap 3 in order to free up time to work on Bootstrap 4. Over 4,000 commits have been made to the Bootstrap 4 codebase so far.

Bootstrap 4 is almost a complete rewrite from Bootstrap 3. Significant changes include:

- Switched from Less to Sass.
- Dropped IE8, IE9, and iOS 6 support.
- Added flexbox support and then dropped non-flexbox support.

- Switched from pixels to root ems.
- Increased global font-size from 14px to 16px.
- Dropped the panel, thumbnail, and well components in favor of cards.
- Dropped the Glyphicons icon font.
- Dropped the pager component.
- Rewrote almost all components, jQuery plugins and documentation.

2.4.1 Why Bootstrap?

Easy to Use

It is extremely an easy and speedy procedure to begin with Bootstrap. Bootstrap is very adaptable too. You can utilize Bootstrap along with CSS, or LESS, or also with Sass

Responsiveness

Every year mobile devices persist to grow hugely popular, and the requirement to have a responsive website has become compulsory and important too. As the fluid grid layout amends vigorously to the appropriate screen resolution, thus crafting a mobile-ready site is a smooth and easy task along with Bootstrap.

The Speed of the Development

One of the main benefits of utilizing Bootstrap happens to be the speed of the development. While driving out a new, fresh website or application swiftly, you should certainly reflect upon utilizing Bootstrap. Instead of coding from scrape, Bootstrap lets you to use ready-made coding blocks in order to assist you in setting up. You can even buy ready-made Bootstrap themes and alter them to fit your requirements, for gaining the quickest potential route.

Customizable Bootstrap

The Bootstrap can be customized as per the designs of your project. The web developers can make a choice to select the aspects which are required which can be simply complete by utilizing Bootstrap customize page. You just have to tick off all the aspects that you do not require, such as- Common CSS: typography, code, grid system, tables, buttons, forms, print media styles; Components: input groups, button groups, pager, labels, navs, navbar, badges, pagination; JavaScript components:

dropdowns, popovers, modals, tooltips, carousels; Utilities: Responsive utilities, basic utilities. Thus your custom version of Bootstrap is all set for download process.

Consistency

Few Twitter employees firstly expanded Bootstrap as a framework for boosting the consistency across interior tools. But later the Co-founder Mark Otto after understanding the actual potential released in August 2011 the first open-source version of Bootstrap. He even portrayed how the Bootstrap was enlarged with the use of one core concept- pairing of designers along with developers. Thus Bootstrap became popular on Twitter.

Support

As Bootstrap holds a big support community, you can be provided with help whenever there comes any problem. The creators always keep the Bootstrap updated. Presently Bootstrap is hosted, expanded, and preserved on the GitHub along with more than 9,000 commits, as well as more than 500 contributors.

Packaged JavaScript Components

Bootstrap approaches with a pack of JavaScript components for including the functionality that crafts it in simple way for operating things, such as tooltips, modal windows, alerts, etc. You can even leave out the writing scripts completely.

Simple Integration

Bootstrap can be simply integrated along with distinct other platforms and frameworks, on existing sites and new ones too. You can also utilize particular elements of Bootstrap along with your current CSS.

Grid

Bootstrap has the capability to utilize a 12-column grid that is responsive. It also upholds offset and nested elements. The grid can be maintained in a responsive mode, or you can simply modify it to a secured layout.

Pre-styled Components

Bootstrap approaches with pre-styled components for alerts, dropdowns, nav bars, etc. Hence, being a feature-rich, Bootstrap provides numerous advantages of using it. Hope you would have

understood the above reasons so that you can easily use Bootstrap for making superb web designs for your sites.

2.5 HTML



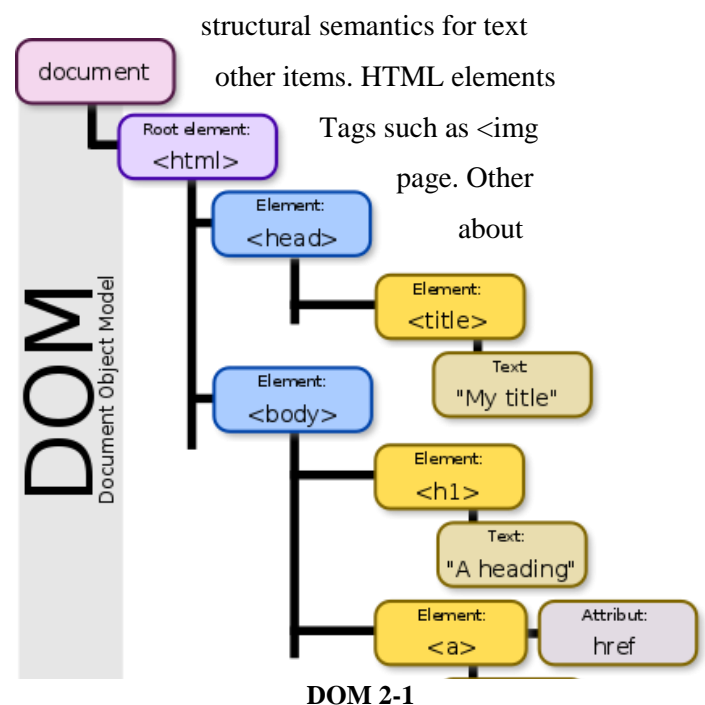
HTML 2-1

“**Hypertext Markup Language (HTML)** is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.”

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting such as headings, paragraphs, lists, links, quotes and are delineated by tags, written using angle brackets. `</>` and `<input />` directly introduce content into the tags such as `<p>` surround and provide information document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

“An HTML element is an individual component of an HTML document or web page, once this has been parsed into the Document Object Model. HTML is composed of a tree of HTML nodes, such as text nodes. Each node can have HTML attributes specified. Nodes can also have content,



DOM 2-1

including other nodes and text. Many HTML nodes represent semantics, or meaning. For example, the <title> node represents the title of the document.”

“The Document Object Model (DOM) is a cross-platform and language-independent application programming interface that treats an HTML, XHTML, or XML document as a tree structure wherein each node is an object representing a part of the document. The objects can be manipulated programmatically and any visible changes occurring as a result may then be reflected in the display of the document.”

The principal standardization of DOM was handled by the World Wide Web Consortium.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

2.6 CSS



CSS 2-1

“**Cascading Style Sheets (CSS)** is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.”

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-

based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C).

Syntax

CSS has a simple syntax and uses a number of English keywords to specify the names of various style properties.

A style sheet consists of a list of rules. Each rule or rule-set consists of one or more selectors, and a declaration block.

Selector

In CSS, selectors declare which part of the markup a style applies to by matching tags and attributes in the markup itself.

Selectors may apply to:

- all elements of a specific type, e.g. the second-level headers h2
- elements specified by attribute, in particular:
 - id: an identifier unique within the document
 - class: an identifier that can annotate multiple elements in a document
- elements depending on how they are placed relative to others in the document tree.

Classes and IDs are case-sensitive, start with letters, and can include alphanumeric characters and underscores. A class may apply to any number of instances of any elements. An ID may only be applied to a single element. Selectors may be combined in many ways to achieve great specificity and flexibility.

```
h1 { color: white;
background: orange;
border: 1px solid black;
padding: 0 0 0 0;
font-weight: bold;
}
/* begin: seaside-theme */

body {
background-color:white;
color:black;
font-family:Arial,sans-serif;
margin: 0 4px 0 0;
border: 12px solid;
}
```

CSS

CSS 2-2

Less (stylesheet language)

“Less (sometimes stylized as LESS) is a dynamic preprocessor style sheet language that can be compiled into Cascading Style Sheets (CSS) and run on the client side or server side. The indented syntax of Less is a nested metalanguage, as valid CSS is valid Less code with the same semantics. Less provides the following mechanisms: variables, nesting, mixins, operators and functions; the main difference between Less and other CSS precompilers being that Less allows real-time compilation via less.js by the browser.”



LESS 2-1

Sass (stylesheet language)

“Sass is a preprocessor scripting language that is interpreted or compiled into Cascading Style Sheets (CSS). SassScript is the scripting language itself. Sass consists of two syntaxes. The original syntax, called "the indented syntax", uses a syntax similar to Haml. It uses indentation to separate code blocks and newline characters to separate rules. The newer syntax, "SCSS" (Sassy CSS), uses block formatting like that of CSS. It uses braces to denote code blocks and semicolons to separate lines within a block. The indented syntax and SCSS files are traditionally given the extensions .sass and .scss, respectively. SassScript provides the following mechanisms: variables, nesting, mixins, and selector inheritance.”



SASS 2-1

2.7 JavaScript



JavaScript

JavaScript 2-1

“JavaScript ("JS" for short) is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites. It was invented by

Brendan Eich, co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation.”

JavaScript was initially created to “make webpages alive”.

The programs in this language are called scripts. They can be written right in the HTML and execute automatically as the page loads.

Scripts are provided and executed as a plain text. They don’t need a special preparation or a compilation to run.

When JavaScript was created, it initially had another name: “LiveScript”. But Java language was very popular at that time, so it was decided that positioning a new language as a “younger brother” of Java would help.

But as it evolved, JavaScript became a fully independent language, with its own specification called ECMAScript, and now it has no relation to Java at all.

At present, JavaScript can execute not only in the browser, but also on the server, or actually on any device where exists a special program called the JavaScript engine.

The browser has an embedded engine, sometimes it’s also called a “JavaScript virtual machine”.

Different engines have different “codenames”, for example:

- V8 – in Chrome and Opera.
- Gecko – in Firefox.
- ...There are other codenames like “Trident”, “Chakra” for different versions of IE, “ChakraCore” for Microsoft Edge, “Nitro” and “SquirrelFish” for Safari etc.

How engines work?

Engines are complicated. But the basics are easy.

The script is written and distributed as a plain text (can be compressed/optimized by so-called “javascript minifiers”).

The engine (embedded if it’s a browser) reads the script (“parses”) and converts (“compiles”) it to the machine language. And then it runs, pretty fast.

The engine applies optimizations on every stage of the process. It even watches the script as it runs, analyzes the data which flows through it and applies optimizations to the machine-code basing on that knowledge. That's why the code runs fast.

What can in-browser JavaScript do?

The modern JavaScript is a “safe” programming language. It does not provide low-level access to memory or CPU, because it was initially created for browsers which do not require it.

The capabilities greatly depend on the environment which runs JavaScript. For instance, Node.JS supports functions that allows JavaScript to read/write arbitrary files, perform network requests etc.

In-browser JavaScript can do everything related to webpage manipulation, interaction with the user and the webserver.

For instance, in-browser JavaScript is able to:

- Add new HTML to the page, change the existing content, modify styles.
- React on user actions, run on mouse clicks, pointer movements, key presses.
- Send requests over the network to remote servers, download and upload files (so-called AJAX and COMET technologies).
- Get and set cookies, ask questions to the visitor, show messages.
- Remember the data on the browser side (“local storage”).

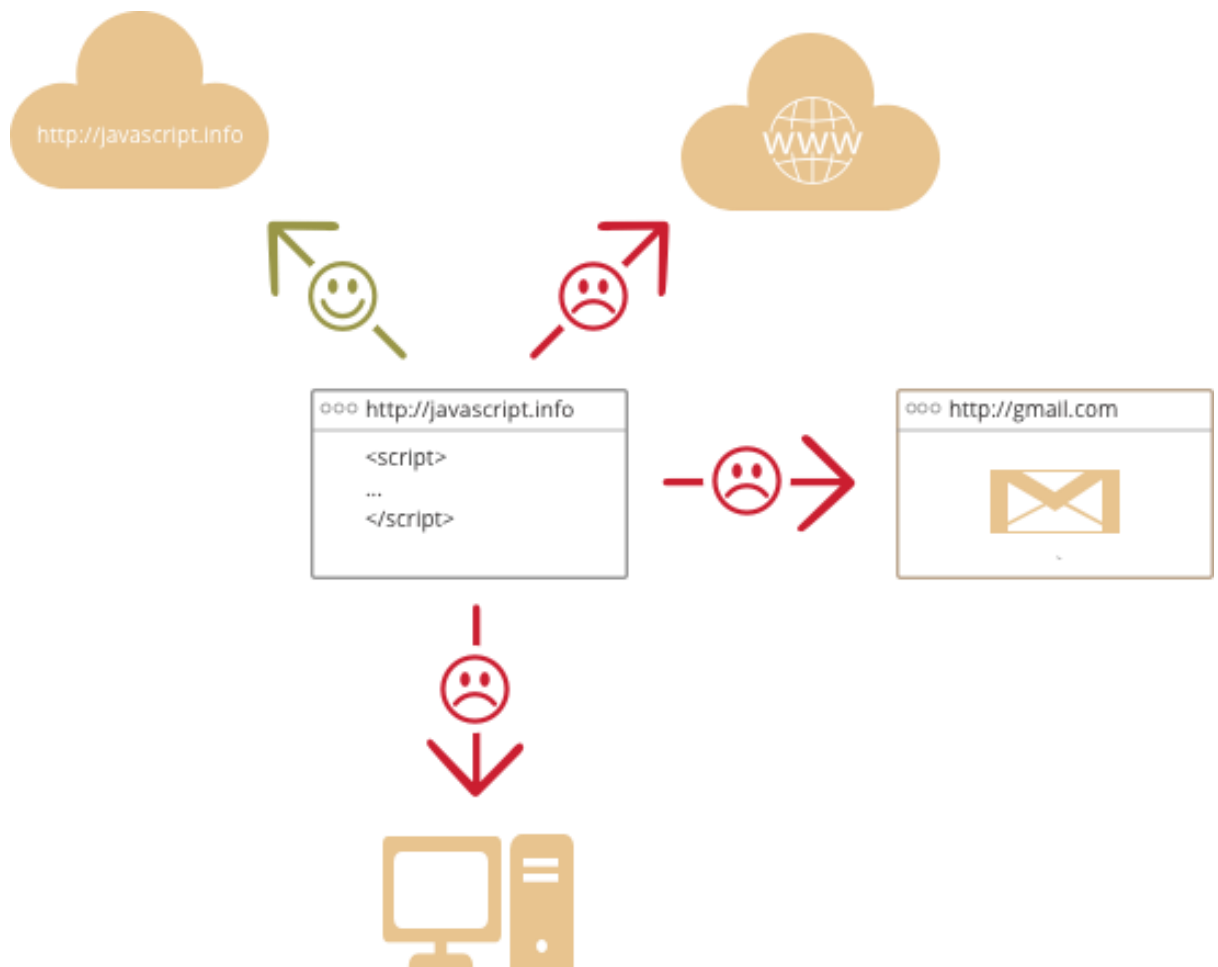
What in-browser JavaScript can NOT do?

JavaScript abilities in the browser are limited for the sake of the user's safety. The aim is to prevent an evil webpage from accessing private information or harming the user's data.

The examples of such restrictions are:

- JavaScript on the webpage may not read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS system functions.
- Modern browsers allow it to work with files, but the access is limited and only provided if the user does certain actions, like “dropping” a file into a browser window or selecting it via an `<input>` tag.

- There are ways to interact with camera/microphone and other devices, but they require a user's explicit permission. So a JavaScript-enabled page may not sneakily enable a web-camera, observe the surroundings and send the information to the NSA.
- Different tabs/windows generally do not know about each other. Sometimes they do, for example when one window uses JavaScript to open the other one. But even in this case, JavaScript from one page may not access the other if they come from different sites (from a different domain, protocol or port).
- That is called a "Same Origin Policy". To work around that, both pages must contain a special JavaScript code that handles data exchange.
- The limitation is again for user's safety. A page from <http://anysite.com> which a user has opened occasionally must not be able to open or access another browser tab with the URL <http://gmail.com> and steal information from there.
- JavaScript can easily communicate over the net to the server where the current page came from. But its ability to receive data from other sites/domains is crippled. Though possible, it requires the explicit agreement (expressed in HTTP headers) from the remote side. Once again, that's safety limitations. Such limits do not exist if JavaScript is used outside of the browser, for example on a server. Modern browsers also allow installing plugin/extensions which may get extended permissions.



JavaScript Limitations 2-1

What makes JavaScript unique?

There are at least three great things about JavaScript:

- Full integration with HTML/CSS.
- Simple things done simply.
- Supported by all major browsers and enabled by default.

Combined, these 3 things exist only in JavaScript and no other browser technology.

That's what makes JavaScript unique. That's why it's the most widespread tool to create browser interfaces.

While planning to learn a new technology, it's beneficial to check its perspectives. So let's move on to the modern trends that include new languages and browser abilities.

Languages “over” JavaScript

The syntax of JavaScript does not suit everyone’s needs. Different people want different features. That’s normal, because projects and requirements are different for everyone.

So recently a plethora of new languages appeared, which are transpiled (converted) to JavaScript before they run in the browser.

The modern tools make the transpilation very fast and transparent, actually allowing developers to code in another language, autoconverting it “under the hood”.

Examples of such languages:

- CoffeeScript is a “syntax sugar” for JavaScript, it introduces shorter syntax, allowing to write more precise and clear code. Usually Ruby guys like it.
- TypeScript is concentrated on adding “strict data typing”, to simplify development and support of complex systems. It is developed by Microsoft.
- Dart is a standalone language that has its own engine that runs in non-browser environments (like mobile apps). It was initially offered by Google as a replacement for JavaScript, but as of now, browsers require it to be transpiled to JavaScript just like the ones above.

There are more. Of course even if we use one of those languages, we should also know JavaScript, to really understand what we’re doing.

Summary

JavaScript was initially created as a browser-only language, but now it is used in many other environments as well.

At this moment, JavaScript has a unique position as a most widely adopted browser language with full integration with HTML/CSS.

There are many languages that get “transpiled” to JavaScript and provide certain features. It is recommended to take a look at them, at least briefly, after mastering JavaScript.

2.7.1 jQuery



jQuery 2-1

“jQuery is defined as a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is not a programming language instead it is a cross-platform JavaScript library.”

What is AJAX?

AJAX = Asynchronous JavaScript And XML.

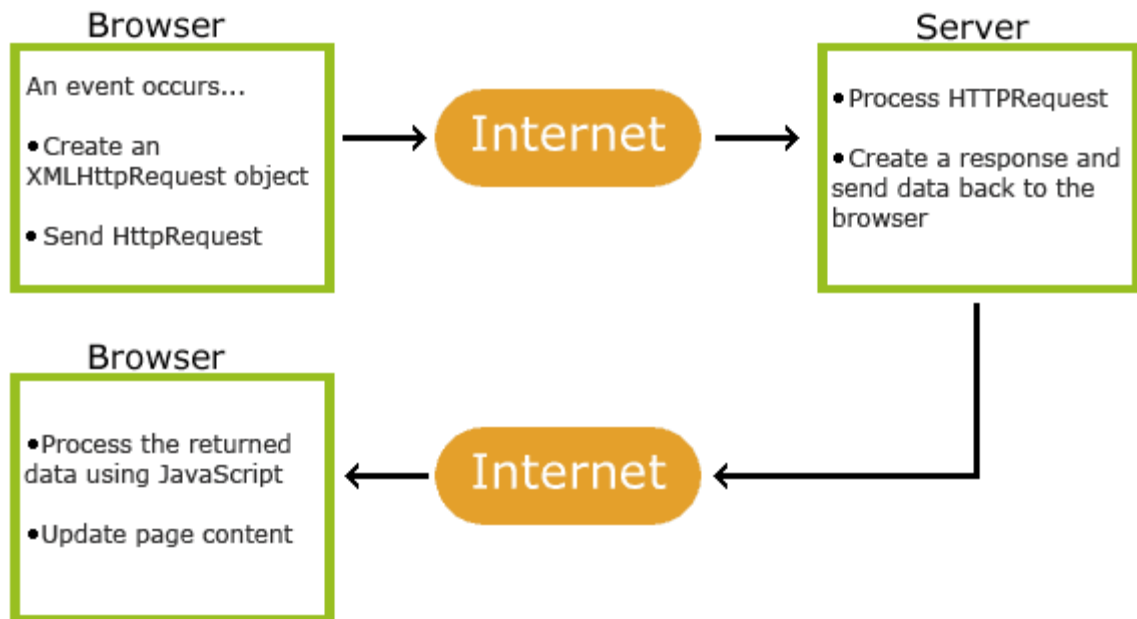
AJAX is not a programming language.

AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
 - JavaScript and HTML DOM (to display or use the data)
- Differences between JavaScript and jQuery

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

How AJAX Works



How AJAX works 2-1

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

Differences between JavaScript and jQuery

The main difference between Javascript and jQuery is that JavaScript is a programming language, while jQuery is a library.

JavaScript allows you to make everything from a programming language perspective: functions, loops, data structures and all that it's paradigm involves.

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an

easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

In general, for a basic comprehension, jQuery uses resources given by JavaScript to make things easier while manipulating DOM and a lot of other things (Ajax, JSON, events).

For jQuery, there are a lot of other libraries available. Everything you do with jQuery, can be obtained using JavaScript, but you will have to write much more code. There exist ups and downs that you must evaluate.

Summary

JavaScript is the programming language of HTML and the Web. It defines the behavior of a web page.

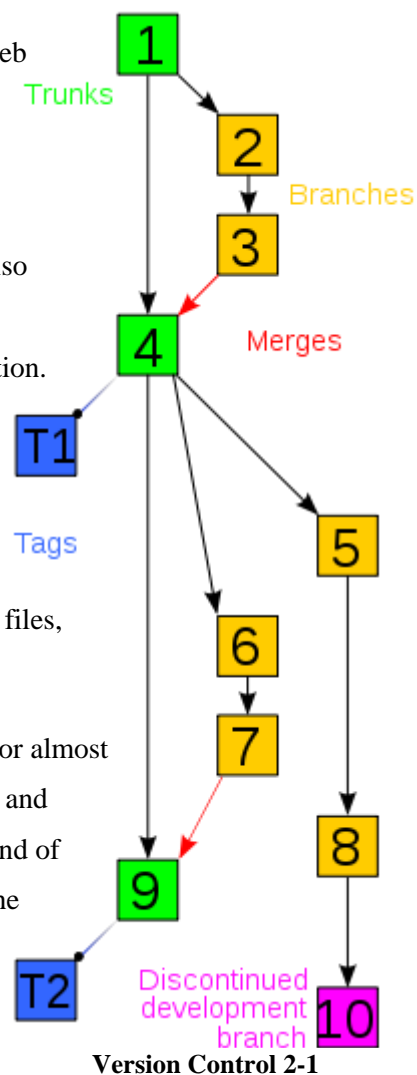
jQuery is a JavaScript Library which greatly simplifies JavaScript programming.

AJAX is the art of exchanging data with a server, and update parts of a web page - without reloading the whole page.

2.8 Version Control

“A component of software configuration management, version control, also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information. Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.”

The need for a logical way to organize and control revisions has existed for almost as long as writing has existed, but revision control became much more important, and complicated when the era of computing began. The numbering of book editions and of specification revisions are examples that date back to the print-only era. Today, the most capable (as well as complex) revision control systems are those used in software development, where a team of people may change the same files.



Version control systems (VCS) most commonly run as stand-alone applications, but revision control is also embedded in various types of software such as word processors and spreadsheets, collaborative web docs and in various content management systems, e.g., Wikipedia's page history. Revision control allows for the ability to revert a document to a previous revision, which is critical for allowing editors to track each other's edits, correct mistakes, and defend against vandalism and spamming.

Software tools for version control are essential for the organization of multi-developer projects

2.8.1 GIT

“**Git** is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.”



GIT 2-1

Git has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

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, but it can be used to keep track of changes in any set of files. As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano.

As with most other distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server.

Git is free and open source software distributed under the terms of the GNU General Public License version 2.

“**GitHub** Inc. is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its



GitHub 2-1

own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.”

GitHub offers plans for both private repositories and free accounts which are commonly used to host open-source software projects. As of June 2018, GitHub reports having over 28 million users and 57 million repositories (including 28 million public repositories), making it the largest host of source code in the world.

3 REQUIREMENTS AND APPLICATION SCHEME

3.1 Requirements

The application is a Web Application designed to detect and recognize human emotions in images. The final output after every image analysis is represented either by the emotions and the details of each face detected in the image or by some error messages if an error occurs.

The application takes an image as an input and returns the confidence across a set of emotions for each face in the image, as well as a bounding box for the face from the Face API. The emotions detected are happiness, sadness, surprise, anger, fear, contempt, disgust, or neutral. These emotions are communicated cross-culturally and universally via the same basic facial expressions, where are identified by the application. There will be also displayed other face details such as: hair details, makeup details, face accessories, age, gender, smile.

In interpreting results from the Face API, the main emotion detected should be interpreted as the emotion with the highest score, as scores are normalized to sum to one.

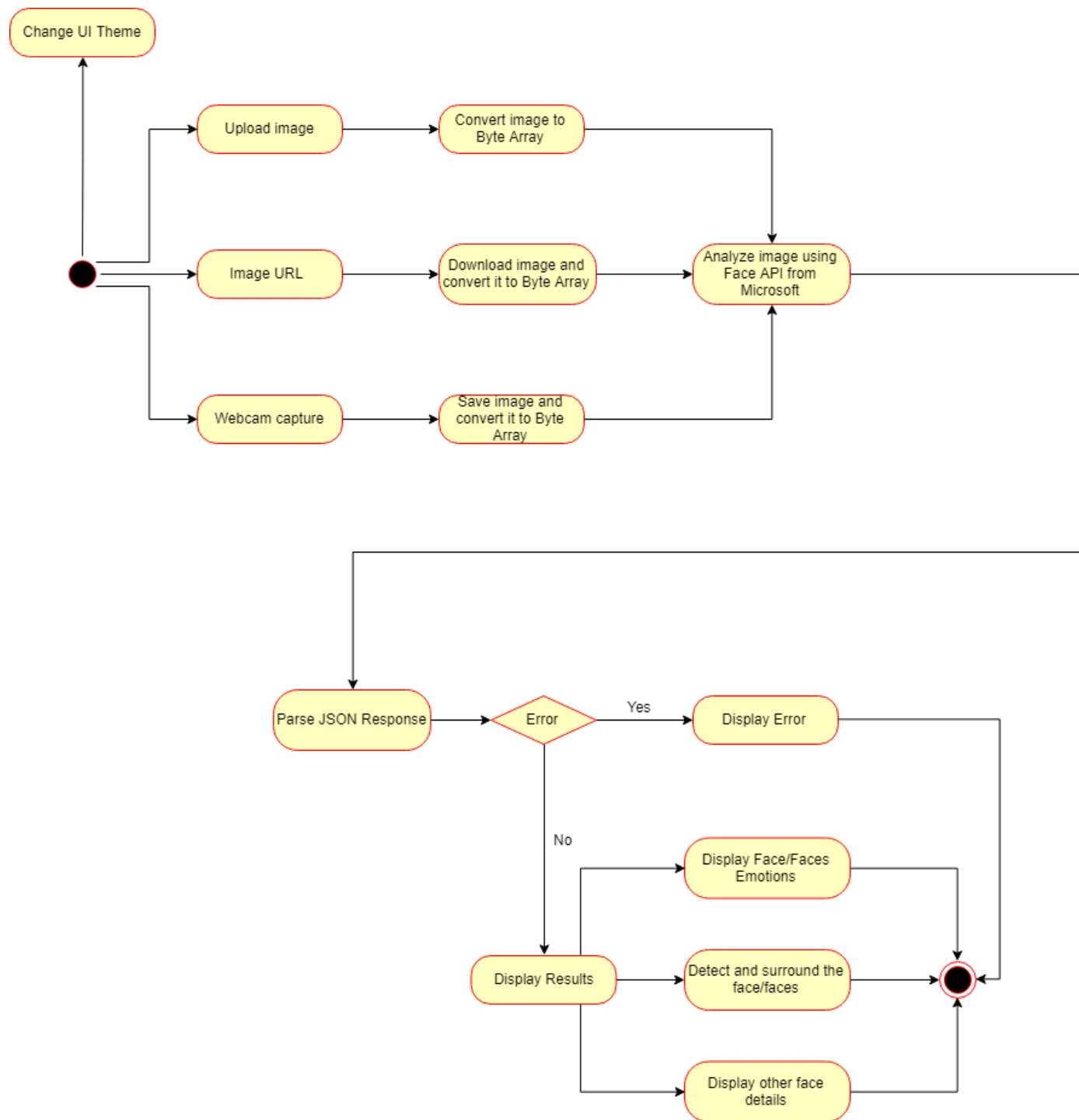
3.2 Functional requirements

General project requirements

- Front-end
 - Upload image from disk and preview it
 - Upload image from URL and preview it
 - Open Webcam
 - Take Webcam capture and preview it

- Close Webcam
 - Change the current image
 - Change UI theme
 - Draw faces rectangles base on faces positions
 - Display either success results or errors
- Back-end
 - Analyze the image and return either a success or an error response

3.3 Application flow



Application Flow 3-1

4 APPLICATION DEVELOPMENT

4.1 Choosing Face API Microsoft

The image analysis could have been done either by using a c++ / python library such as dlib, OpenCv etc. or by using an API.

I chose to use an API because this implies a cleaner solution, less code and a more elegant solution.

How Do Emotion Recognition APIs Work?

With facial emotion detection, algorithms detect faces within a photo or video, and sense micro expressions by analyzing the relationship between points on the face, based on curated databases compiled in academic environments.

To detect emotion in the database, sentiment analysis processing software can analyze text to conclude if a statement is generally positive or negative based on keywords and their valence index. Lastly, sonic algorithms have been produced that analyze recorded speech for both tone and word content.

There are already many Emotion APIs that can be used for developing applications which imply Emotion Recognition / Detection such as: Face API from Microsoft, Emotient, Affectiva, EmoVu, Nviso, Kairos, Project Oxford by Microsoft, Face Reader by Noldus, Sightcorp, SkyBiometry, Face++, Imotions, CrowdEmotion, FacioMetrics, IBM Watson, Receptiviti, AlchemyAPI, Bitext, Mood Patrol, Synesketech, Tone API, Repustate API, Good Vibrations, Vokaturi etc.

Machine emotional intelligence is still evolving, but the future could soon see targeted ads that respond to not only our demographic (age, gender, likes, etc.) but to our current emotional state. For point of sale advertising, this information could be leveraged to nudge sales when people are most emotionally vulnerable, getting into some murky ethical territory. Emotional recognition via facial detection is also shady if the user isn't aware of their consent to be recorded visually. There are of course data privacy legalities any API provider or consumer should be aware of before implementation.

We are only on the tip of the iceberg when it comes to machine human interaction, but cognitive computing technologies like these are exciting steps toward creating true machine emotional intelligence.

I chose to use Face API from Microsoft because it was the closest to my needs, it is optimized and easy to use. Typically, it remembers stateful information for the stateless HTTP protocol.

4.2 Front end

4.2.1 Change UI theme

Changing the UI theme is done by saving the theme name using a cookie.

An HTTP cookie (web cookie, browser cookie) is a small piece of data that a server sends to the user's web browser. The browser may store it and send it back with the next request to the same server.

Cookies are mainly used for three purposes:

- Session management: Logins, shopping carts, game scores, or anything else the server should remember
- Personalization: User preferences, themes, and other settings
- Tracking: Recording and analyzing user behavior

Cookies are included in the following categories:

- Session cookies: it is deleted when the client shuts down
- Permanent cookies: instead of expiring when the client closes, permanent cookies expire at a specific date
- Per page: for only one page of the application
- Per application: for all pages of the application

For this functionality I used a cookie for all pages in the application which saves the name of the selected theme into the browser. When the application loads the value is retrieved from the cookie (if it exists) and the corresponding CSS bundle is loaded in the Layout section of the application.

Only two themes were included for demonstration purposes:

- Bootstrap-minty

Emotional recognition About

bootstrap-minty

Select Photo

Upload a photo in order to analyze it.

Change Image Image URL

Webcam

Image analysis results

Detection result:
1 face detected
The main emotion seems to be: neutral

```
{  
  Anger: 0.007  
  Contempt: 0.016  
  Disgust: 0.004  
  Fear: 0  
  Happiness: 0.404  
  Neutral: 0.568  
  Sadness: 0.001  
  Surprise: 0  
}
```

Name	Value	Domain	Path	Expires / ...	Size	HTTP	Secur
1P_JAR	2018-06-24-06	.gstatic.com	/	2018-07-2...	19		
CONSENT	WP.26e7c7	.gstatic.com	/	2038-01-0...	16		
NID	133=JbbQoFXHDTvez-EkPScXBIM1kb7mog_EnwXgQk4ahy8FBfG7...	.gstatic.com	/	2018-12-2...	135	✓	
bootstrapCss	bootstrap-minty	localhost	/	1969-12-3...	27		

Bootstrap theme 4-1

- Bootstrap-darkly

Emotional recognition
About

bootstrap-darkly

Select Photo

Upload a photo in order to analyze it.

Change Image
Image URL

Webcam




Image analysis results

Detection result:
1 face detected
The main emotion seems to be: neutral

```
{
  Anger: 0.007
  Contempt: 0.016
  Disgust: 0.004
  Fear: 0
  Happiness: 0.404
  Neutral: 0.568
  Sadness: 0.001
  Surprise: 0
}
```

Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits

Session Storage
IndexedDB
Web SQL
Cookies
http://localhost:60274
Cache

Filter

Name	Value	Domain	Path	Expires / ...	Size	HTTP	Sec
1P_JAR	2018-06-24-06	.gstatic.com	/	2018-07-2...	19		
CONSENT	WP.26e7c7	.gstatic.com	/	2038-01-0...	16		
NID	133=JbbQoFXHDTvez-EkPScXBIM1kb7mog_EnwXgQk4ahy8F8fG7...	.gstatic.com	/	2018-12-2...	135	✓	
bootstrapCss	bootstrap-darkly	localhost	/	1969-12-3...	28		

Bootstrap theme 4-2

4.2.2 Upload image

The application allows to users either to upload an image from the disk or upload an image from an URL.

When uploading an image from disk the image is sent to the Image Controller through an AJAX Post. In the image controller the picture is converted into a byte array and it is sent to the Face API for the analysis part.

JavaScript AJAX call

```
var file = document.getElementById(config.inputImageId).files[0];
if (blob !== null && blob !== undefined)
    file = blob;
var formData = new FormData();
formData.append("uploadedImage", file);
$.ajax({
    url: "/Image/AnalyzeImage",
    type: 'POST',
    data: formData,
    contentType: false,
    processData: false,
    success: function (result) {
        getDetectedFacesPositions();
        document.getElementById(config.emotionSectionId).innerHTML = result;
        document.getElementById(config.emotionSectionId).hidden = false;
        hideLoader();
    },
    error: function (e) {
        document.getElementById(config.emotionSectionId).innerHTML = e.pr;
        document.getElementById(config.emotionSectionId).hidden = false;
        console.log(e);
        hideLoader();
    }
});
```

AJAX call 4-1

After uploading an image, the above AJAX post sends the image to the server controller in order to be processed.

- The URL (type: String) is a string containing the URL to which the request is sent.
- The type (default: 'GET', type: String) is an alias for method.
- The data (type: PlainObject or String or Array) represents data to be sent to the server. It is converted to a query string, if not already a string. It's appended to the url for GET-requests.
- The contentType (default: 'application/x-www-form-urlencoded; charset=UTF-8', type: Boolean or String). When sending data to the server, use this content type. Default is "application/x-www-form-urlencoded; charset=UTF-8", which is fine for most cases. If you explicitly pass in a content-type to \$.ajax(), then it is always sent to the server (even if no data is sent). As of jQuery 1.6 you can pass false to tell jQuery

to not set any content type header. Note: The W3C XMLHttpRequest specification dictates that the charset is always UTF-8; specifying another charset will not force the browser to change the encoding. Note: For cross-domain requests, setting the content type to anything other than application/x-www-form-urlencoded, multipart/form-data, or text/plain will trigger the browser to send a preflight OPTIONS request to the server.

- `processData` (default: true, type: Boolean). By default, data passed in to the data option as an object (technically, anything other than a string) will be processed and transformed into a query string, fitting to the default content-type "application/x-www-form-urlencoded". If you want to send a DOMDocument, or other non-processed data, set this option to false.
- The success function (PlainObject data, String textStatus, jqXHR jqXHR) represents a callback function that is executed if the request succeeds. Required if dataType is provided but can be null in that case.

On the success callback the following actions are performed:

- **`getDetectedFacesPositions()`**; - this method gets an array of face positions and if no error it draws the face rectangles on the image else an error snack bar is displayed.

```
var getDetectedFacesPositions = function () {
    $.ajax({
        url: "/Image/GetDetectedFacesPositions",
        type: 'GET',
        success: function (result) {
            var array = JSON.parse(result);
            surroundFaces(array);
        },
        error: function (e) {
            console.log(e);
            showSnackBar("Failed to detect faces in image!");
        }
    });
}

var showSnackBar = function (message) {
    var x = document.getElementById("snackbar"); // get the DOM element
    representing the snack bar (div element)
```

```

        x.innerHTML = message; // set the text to be displayed as html of the
element
        x.className = "show"; // set the class to show the snack bar
        setTimeout(function () { x.className = x.className.replace("show", "");
}, 3000); // set a timeout of 3 seconds after which the snack bar is hidden by
removing the “show” CSS class.
    }

```

After analyzing the image, the Face API response also contains an array of face positions. A face position is represented by 4 values: Top (the distance between the top border of the image and the top border of the face), Left (the distance between the left border of the image and the left border of the face), Width (the distance between the left border and the right border of the face) and Height (the distance between the top border and the bottom border of the face). These values are measured in pixels.

In the application the uploaded image is resized to fit the screen but on the server side the image dimensions are the original ones, so all the face positions returned by the server are corresponding to the original size of the image. Regarding this, when the face rectangles are drawn the face dimensions from the server response are scaled to the resized image dimensions in order to perfectly fit the faces detected in the image. After that for each face in the image the rectangle is drawn over the image using a canvas HTML element. The stroke() method actually draws the path you have defined with ctx.rect(left, top, width, height);. The default color is black but I set it to red: ctx.strokeStyle = "#FF0000";

Code:

```

var surroundFaces = function (facesArr) {
    var uploadedImage =
document.getElementById(config.uploadedImageId);
    var imageOriginalWidth = uploadedImage.naturalWidth;
    var imageOriginalHeight = uploadedImage.naturalHeight;

    var widthScale = imageOriginalWidth / uploadedImage.width;
    var heightScale = imageOriginalHeight / uploadedImage.height;

    var canvas = document.getElementById(config.imageCanvasId);
    var ctx = canvas.getContext("2d");
    canvas.style.bottom = uploadedImage.height + "px";

```

```
ctx.canvas.width = uploadedImage.width;
ctx.canvas.height = uploadedImage.height;
```

```
for (var index in facesArr) {
    var top = parseInt(facesArr[index].Top / heightScale);
    var left = parseInt(facesArr[index].Left / widthScale);
    var width = parseInt(facesArr[index].Width / widthScale);
    var height = parseInt(facesArr[index].Height / heightScale);
    ctx.rect(left, top, width, height);
}
```

```
ctx.strokeStyle = "#FF0000";
ctx.stroke();
document.getElementById(config.uploadedImageId).style.height =
uploadedImage.height + "px !important";
canvas.hidden = false;
}
```

- document.getElementById(config.emotionSectionId).innerHTML = result;

The partial view's html is replaced with the html returned by the controller after the image was analyzed.

- document.getElementById(config.emotionSectionId).hidden = false;

Then the content is displayed.

- hideLoader();

Finally, the loader is hidden

- The error callback is a function (jqXHR jqXHR, String textStatus, String errorThrown) to be called if the request fails.

If the request fails:

- The error is displayed instead of the result of the analysis.

```
document.getElementById(config.emotionSectionId).innerHTML = e.pr;
```

- The corresponding area is shown

```
document.getElementById(config.emotionSectionId).hidden = false;
```

- The error is also printed in the browser console

```
console.log(e);
```

- Finally, the loader is hidden

hideLoader();

The **showLoader()** method is used to display a CSS made loader which informs the user that an action is in progress and it is not finished yet and also to disable the HTML elements which are related to that action. Code:

```
var showLoader = function () {
    document.getElementById(config.loaderDivId).hidden = false;
    document.getElementById(config.analyzePhotoBtnId).disabled = true;
    document.getElementById(config.showImageFromUrlBtnId).disabled = true;
    document.getElementById(config.chooseImageBtnId).disabled = true;
    document.getElementById(config.analyzePhotoBtnId).classList.add(config.customDisabledBtnClass);
    document.getElementById(config.showImageFromUrlBtnId).classList.add(config.customDisabledBtnClass);
    document.getElementById(config.chooseImageBtnId).classList.add(config.customDisabledBtnClass);
    document.getElementById(config.imageUrlId).disabled = true;
    var getMoreInfoAboutPersonsBtn = document.getElementById(config.getMoreInfoAboutPersonsBtnId);
    if (getMoreInfoAboutPersonsBtn !== null && getMoreInfoAboutPersonsBtn !== undefined) {
        getMoreInfoAboutPersonsBtn.disabled = true;
        getMoreInfoAboutPersonsBtn.classList.add(config.customDisabledBtnClass);
    }
    document.getElementById(config.takeSnapshotBtnId).disabled = true;
    document.getElementById(config.takeSnapshotBtnId).classList.add(config.customDisabledBtnClass);
    document.getElementById(config.openWebcamBtnId).disabled = true;
    document.getElementById(config.openWebcamBtnId).classList.add(config.customDisabledBtnClass);
    document.getElementById(config.stopWebcamBtnId).disabled = true;
    document.getElementById(config.stopWebcamBtnId).classList.add(config.customDisabledBtnClass);
}
```

Show Loader Method 4-1

The **hideLoader()** method is used to hide the displayed loader, informing the user that an action is completed and also to enable the HTML elements which are related to that action. Code:

```
var hideLoader = function () {
    document.getElementById(config.loaderDivId).hidden = true;
    document.getElementById(config.analyzePhotoBtnId).disabled = false;
    document.getElementById(config.showImageFromUrlBtnId).disabled = false;
    document.getElementById(config.chooseImageBtnId).disabled = false;
    document.getElementById(config.analyzePhotoBtnId).classList.remove(config.customDisabledBtnClass);
    document.getElementById(config.showImageFromUrlBtnId).classList.remove(config.customDisabledBtnClass);
    document.getElementById(config.chooseImageBtnId).classList.remove(config.customDisabledBtnClass);
    document.getElementById(config.imageUrlId).disabled = false;
    var getMoreInfoAboutPersonsBtn = document.getElementById(config.getMoreInfoAboutPersonsBtnId);
    if (getMoreInfoAboutPersonsBtn !== null && getMoreInfoAboutPersonsBtn !== undefined) {
        getMoreInfoAboutPersonsBtn.disabled = false;
        getMoreInfoAboutPersonsBtn.classList.remove(config.customDisabledBtnClass);
    }
    document.getElementById(config.takeSnapshotBtnId).disabled = false;
    document.getElementById(config.takeSnapshotBtnId).classList.remove(config.customDisabledBtnClass);
    document.getElementById(config.openWebcamBtnId).disabled = false;
    document.getElementById(config.openWebcamBtnId).classList.remove(config.customDisabledBtnClass);
    document.getElementById(config.stopWebcamBtnId).disabled = false;
    document.getElementById(config.stopWebcamBtnId).classList.remove(config.customDisabledBtnClass);
}
```

Hide Loader 4-1

Controller action (will be explained in the section regarding the image analysis)

```
[HttpPost]
public async Task<ActionResult> AnalyzeImage(HttpPostedFileBase uploadedImage)
{
    try
    {
        byte[] fileData = null;
        using (var binaryReader = new BinaryReader(uploadedImage.InputStream))
        {
            fileData = binaryReader.ReadBytes(uploadedImage.ContentLength);
        }
        var emotions = await MakeAnalysisRequest(fileData);
        return PartialView("~/Views/Image/_EmotionSection.cshtml", emotions);
    }
    catch (Exception ex)
    {
        var emotionModelList = new List<EmotionSectionModel>();
        emotionModelList.Add(new EmotionSectionModel(ex.ToString()));
        return PartialView("~/Views/Image/_EmotionSection.cshtml", emotionModelList);
    }
}
```

Controller Action 4-1

When uploading an image from an URL the URL is sent to the Image Controller through an AJAX Post. In the image controller the picture is downloaded and converted into a byte array and then it is sent to the Face API for the analysis part.

JavaScript AJAX call

```
$.ajax({
    url: "/Image/AnalyzeImageFromUrl",
    type: 'POST',
    data: { imageUrlAddress },
    success: function (result) {
        getDetectedFacesPositions();
        document.getElementById(config.emotionSectionId).innerHTML = result;
        document.getElementById(config.emotionSectionId).hidden = false;
        hideLoader();
    },
    error: function (e) {
        document.getElementById(config.emotionSectionId).innerHTML = e;
        document.getElementById(config.emotionSectionId).hidden = false;
        console.log(e);
        hideLoader();
    }
});
```

AJAX call 4-2

Controller action (will be explained in the section regarding the image analysis)

```
[HttpPost]
public async Task<ActionResult> AnalyzeImageFromUrl(string imageUrlAddress)
{
    try
    {
        byte[] fileData = GetImageAsByteArrayFromUrl(imageUrlAddress);
        var emotions = await MakeAnalysisRequest(fileData);
        return PartialView("~/Views/Image/_EmotionSection.cshtml", emotions);
    }
    catch (Exception ex)
    {
        var emotionModelList = new List<EmotionSectionModel>();
        emotionModelList.Add(new EmotionSectionModel(ex.ToString()));
        return PartialView("~/Views/Image/_EmotionSection.cshtml", emotionModelList);
    }
}

static byte[] GetImageAsByteArrayFromUrl(string imageFilePath)
{
    var webClient = new WebClient();
    byte[] imageBytes = webClient.DownloadData(imageFilePath);
    return imageBytes;
}
```

Controller Action 4-2

4.2.3 Webcam capture

The main functions of the webcam are implemented through JavaScript.

For managing the webcam actions, the `Navigator.getUserMedia()` method is used.

The `Navigator.getUserMedia()` method prompts the user for permission to use up to one video input device (such as a camera or shared screen) and up to one audio input device (such as a microphone) as the source for a `MediaStream`.

If an error occurs (webcam not detected, the navigator library is not supported by the browser) an error snack bar is displayed.

- Opening the webcam: When clicking the “open webcam” button the video DOM element (“<video id="video" hidden="true" style="margin-top: 10px"></video>”) is used to play the Webcam using the `Navigator.getUserMedia()` method.

A snack bar is shown displaying an error message both if an error occurs while opening the Webcam and if the browser does not support the navigator library is not supported by the browser.

```
$('#' + config.openWebcamBtnId).on('click', function () {
    navigator.getUserMedia = navigator.getUserMedia ||
        navigator.webkitGetUserMedia ||
        navigator.mozGetUserMedia;

    if (navigator.getUserMedia) {
        navigator.getUserMedia({ audio: true, video: { width: 1280, height: 720 } },
            function (stream) {
                var video = document.querySelector(config.videoId);
                showWebcamRelatedButtons();
                video.hidden = false;
                video.srcObject = stream;
                video.onloadedmetadata = function (e) {
                    video.play();
                };
            },
            function (err) {
                console.log("The following error occurred: " + err.name);
                showSnackBar("Error while opening Webcam!");
                hideWebcamRelatedButtons();
            }
        );
    } else {
        console.log("getUserMedia not supported");
        showSnackBar("getUserMedia not supported by your browser!");
        hideWebcamRelatedButtons();
    }
});
```

Open Webcam 4-1

- Taking a snapshot

When the “Take Snapshot” button is clicked the snapshot() method is invoked and it saves a snapshot of the Webcam: “let imageCpture = new ImageCapture(track)”.

If no error the saved capture is displayed together with a success message and the image capture is saved as a blob (binary large object): “blob = dataUriToBlob(img.src)” else an error message is displayed using the snack bar.

When analyzing the image capture the blob is sent to the server identically, using the same method, like an uploaded image from disk is sent (described above).

```
var snapshot = function () {  
  try {  
    var videoElem = document.getElementById(config.videoId);  
    let stream = videoElem.srcObject;  
    let track = stream.getVideoTracks()[0];  
    let imageCapture = new ImageCapture(track);  
    document.getElementById(config.emotionSectionId).hidden = true;  
    document.getElementById(config.personDetailsSectionId).hidden = true;  
    showSnackBar("Successfully captured image!");  
    hideImageFromUrlElements();  
  
    var scale = 0.7;  
    var canvas = document.createElement("canvas");  
    canvas.width = videoElem.videoWidth * scale;  
    canvas.height = videoElem.videoHeight * scale;  
    canvas.getContext('2d')  
      .drawImage(videoElem, 0, 0, canvas.width, canvas.height);  
  
    var input = document.getElementById(config.inputImageId);  
    input.src = canvas.toDataURL();  
  
    document.getElementById(config.imageCanvasId).hidden = true;  
    var img = document.getElementById(config.uploadedImageId);  
    img.src = input.src;  
    img.hidden = false;  
  
    blob = dataUriToBlob(img.src);  
  }  
  catch (err) {  
    showSnackBar(err);  
  }  
}
```

Take Webcam Snapshot 4-1

- Closing the webcam

When the “Close Webcam” button is clicked the video HTML element is hidden along with the webcam related button (“Take Snapshot”, “Close Webcam”).

The `MediaStream.getTracks()` method returns a list of all `MediaStreamTrack` objects stored in the `MediaStream` object, regardless of the value of the `kind` attribute. The order is not defined, and may not only vary from one browser to another, but also from one call to another.

Every track in the list is stopped “`track.stop()`,” and after that the source object of the video element is set to null.

```
var closeWebcam = function () {  
    var videoElem = document.getElementById(config.videoId);  
    videoElem.hidden = true;  
    hideWebcamRelatedButtons();  
    let stream = videoElem.srcObject;  
    let tracks = stream.getTracks();  
  
    tracks.forEach(function (track) {  
        track.stop();  
    });  
  
    videoElem.srcObject = null;  
};
```

Close Webcam 4-1

Webcam error example:

Select Photo

Upload a photo in order to analyze it.

Browse

Image URL

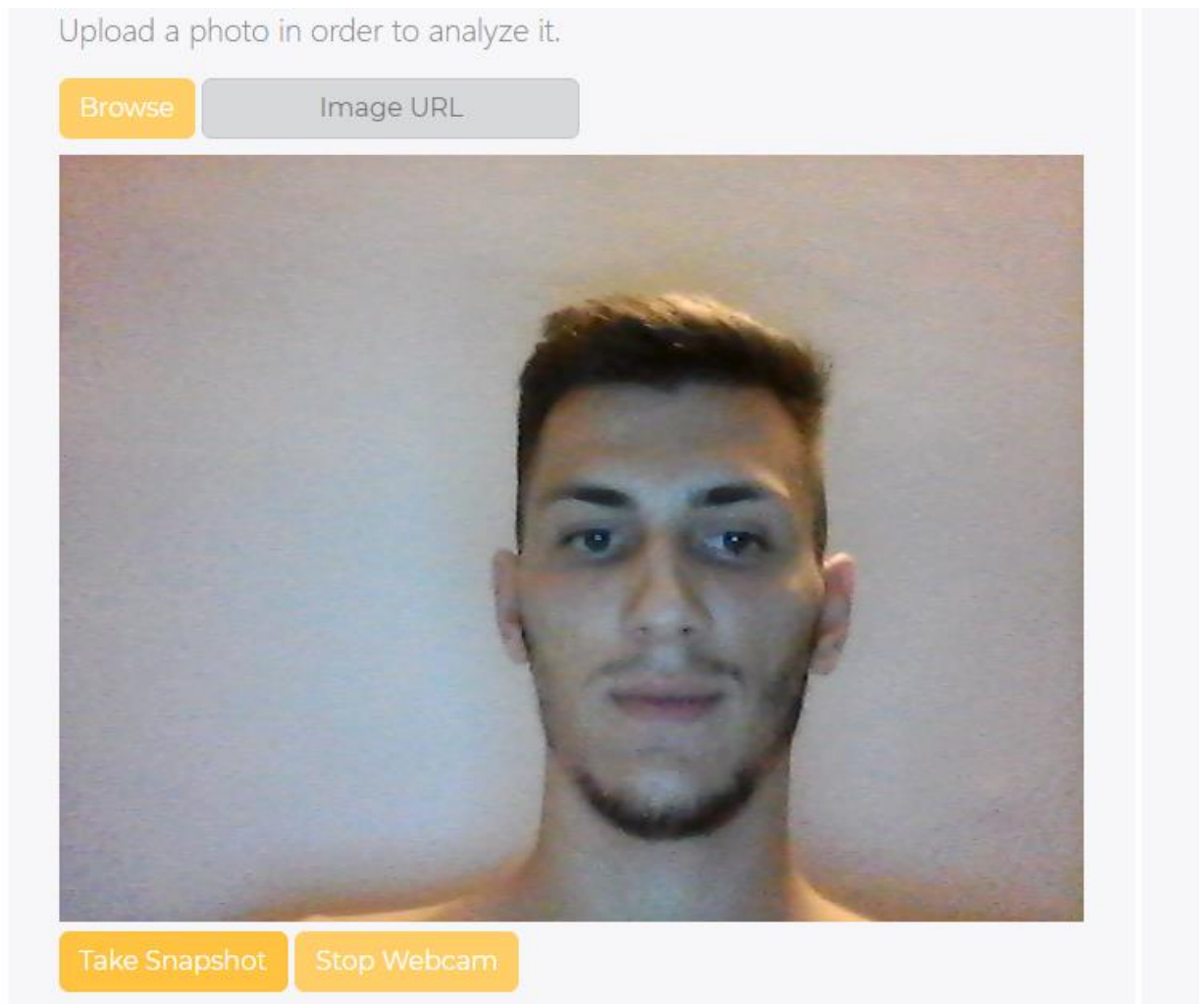
Webcam

:018 - Emotional Recognition Application

getUserMedia not supported by your browser!

Webcam Error 4-1

If no error occurs the live webcam shows up.



Webcam 4-1


4.2.4 Display results

The image analysis results are displayed on the AJAX post success callback (described above) in partial views as scrollable areas returned by the server when the image analysis is completed.

Upload a photo in order to analyze it.

[Change Image](#)

[Webcam](#)



Detection result:
2 faces detected
Person number 1 seems to mainly feel surprise

```
{  
  Anger: 0  
  Contempt: 0  
  Disgust: 0  
  Fear: 0.006  
  Happiness: 0  
  Neutral: 0.002  
  Sadness: 0  
  Surprise: 0.991  
}
```

Person number 2 seems to mainly feel surprise

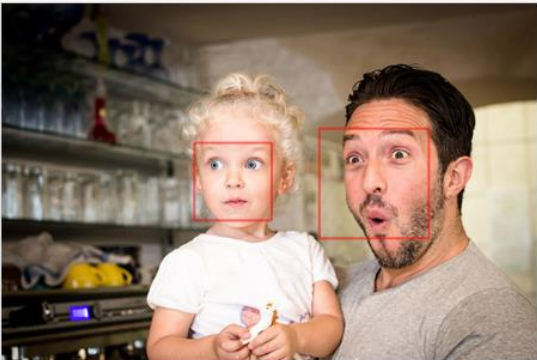
[Get More Info »](#)

Emotions results:

Upload a photo in order to analyze it.

[Change Image](#)

[Webcam](#)



Sadness: 0
Surprise: 0.991

Person number 2 seems to mainly feel surprise

```
{  
  Anger: 0.001  
  Contempt: 0.104  
  Disgust: 0  
  Fear: 0.001  
  Happiness: 0.001  
  Neutral: 0.093  
  Sadness: 0  
  Surprise: 0.8  
}
```

[Get More Info »](#)

Analysis Results 4-2

More face details results

Get More Info »

More details results

Person number 1 details
{
 Gender: male
 Age: 34
 Smile: 0
 Glasses: NoGlasses
 Face Hair
 {
 Moustache: 0.4
 Beard: 0.4
 Sideburns: 0.4
 }
 Makeup
 {
 Eye Makeup: False
 }
}

Get More Info »

More details results

{
 Eye Makeup: False
 Lip Makeup: False
}
 Bald: 0.04
 Hair Colors
 {
 Color: brown - Confidence: 0.98
 Color: black - Confidence: 0.97
 Color: gray - Confidence: 0.33
 Color: other - Confidence: 0.19
 Color: red - Confidence: 0.1
 Color: blond - Confidence: 0.07
 }
}

Person number 2 details

Analysis Results 4-3

Get More Info »

More details results

Person number 2 details
{
 Gender: male
 Age: 3
 Smile: 0.001
 Glasses: NoGlasses
 Face Hair
 {
 Moustache: 0
 Beard: 0
 Sideburns: 0
 }
 Makeup
 {
 Eye Makeup: True
 }
}

Analysis Results 4-5

Analysis Results 4-4

Get More Info »

More details results

{
 Eye Makeup: True
 Lip Makeup: True
}
 Bald: 0.08
 Hair Colors
 {
 Color: blond - Confidence: 1
 Color: gray - Confidence: 0.93
 Color: other - Confidence: 0.26
 Color: brown - Confidence: 0.18
 Color: red - Confidence: 0.06
 Color: black - Confidence: 0.03
 }
}

Analysis Results 4-6

The **“Get More Info” Button** when clicked it performs an ajax GET to the server in order to retrieve more face details about the persons detected in the image.

The extra details are retrieved from the stored image analysis result.

On success a partial view containing the results is displayed.

On error a partial view containing the error stack trace is displayed. A stack trace is a list of the method calls that the application was in the middle of when an Exception was thrown.

The **“Get More Info” Button** is displayed only after an image was analyzed successfully and at least one person was detected in the provided image.

```
var getMoreInfoAboutPersons = function () {  
    showLoader();  
    $.ajax({  
        url: "/Image/GetMoreDetails",  
        type: 'GET',  
        success: function (result) {  
            document.getElementById(config.personDetailsSectionId).innerHTML = result;  
            document.getElementById(config.personDetailsSectionId).hidden = false;  
            hideLoader();  
        },  
        error: function (e) {  
            document.getElementById(config.personDetailsSectionId).innerHTML = e;  
            document.getElementById(config.personDetailsSectionId).hidden = false;  
            console.log(e);  
            hideLoader();  
        }  
    });  
}
```

Get More Info 4-1

For **face surrounding rectangles** a canvas is put over the image and the rectangles are drawn according to the detected face positions. This method was detailed above.

```
var surroundFaces = function (facesArr) {  
    var uploadedImage = document.getElementById(config.uploadedImageId);  
    var imageOriginalWidth = uploadedImage.naturalWidth;  
    var imageOriginalHeight = uploadedImage.naturalHeight;  
  
    var widthScale = imageOriginalWidth / uploadedImage.width;  
    var heightScale = imageOriginalHeight / uploadedImage.height;  
  
    var canvas = document.getElementById(config.imageCanvasId);  
    var ctx = canvas.getContext("2d");  
    canvas.style.bottom = uploadedImage.height + "px";  
    ctx.canvas.width = uploadedImage.width;  
    ctx.canvas.height = uploadedImage.height;  
  
    for (var index in facesArr) {  
        var top = parseInt(facesArr[index].Top / heightScale);  
        var left = parseInt(facesArr[index].Left / widthScale);  
        var width = parseInt(facesArr[index].Width / widthScale);  
        var height = parseInt(facesArr[index].Height / heightScale);  
        ctx.rect(left, top, width, height);  
    }  
  
    ctx.strokeStyle = "#FF0000";  
    ctx.stroke();  
    document.getElementById(config.uploadedImageId).style.height = uploadedImage.height;  
    canvas.hidden = false;  
}
```

Face Rectangles 4-1

4.3 Back end

4.3.1 Analyze Image

Image controller main functions:

The **GetDetectedFacesPositions()** method retrieves all the faces rectangles of each person detected in the image. For each face rectangle it creates a **FacePosition** model populated with the 4 dimensions of the rectangle (Top, Left, Width, Height) and add it to a list. Finally, the list is serialized and sent back to the AJAX GET “GetDetectedFacesPositions”. JSON is a format that encodes objects in a string. Serialization means to convert an object into that string, and deserialization is its inverse operation. When transmitting data or storing them in a file, the data are required to be byte strings, but complex objects are seldom in this format.

```
[HttpGet]
public string GetDetectedFacesPositions()
{
    var facesList = new List<FacePosition>();
    foreach (var item in JsonApiResponse.ToList())
    {
        var faceRectangle = item[Constants.FaceRectangle];
        var facePosition = new FacePosition()
        {
            Top = (int)faceRectangle[Constants.Top],
            Left = (int)faceRectangle[Constants.Left],
            Width = (int)faceRectangle[Constants.Width],
            Height = (int)faceRectangle[Constants.Height]
        };
        facesList.Add(facePosition);
    }

    return JsonConvert.SerializeObject(facesList);
}
```

The **GetMoreDetails()** method retrieves details such as: hair details, makeup details, face accessories, age, gender, smile. The method parses the **JsonApiResponse** and depending on the number of the detected persons it creates a list of “**PersonDetailsModel**”s and returns the partial view “_PersonDetailsSection.cshtml” which will be populated with the data from the **personDetailsModelList**. If an error occurs while parsing the json in order to retrieve the details a new **PersonDetailsModel** is created, only the error field is populated and the partial view “_PersonDetailsSection.cshtml” along with the error model are returned. The “_PersonDetailsSection.cshtml” view only contains HTML displaying the details from its model.

This method is called through the AJAX post to the URL “"/Image/GetMoreDetails"”.

```
[HttpGet]
public ActionResult GetMoreDetails()
{
```

```

var personDetailsModelList = new List<PersonDetailsModel>();
try
{
    if (JsonApiResponse.Count() == 1)
    {
        var personDetailsModel = GetPersonDetailsModel(JsonApiResponse.First());
        personDetailsModelList.Add(personDetailsModel);
        return PartialView("~/Views/Image/_PersonDetailsSection.cshtml", personDetailsModelList);
    }

    if (JsonApiResponse.Count() > 1)
    {
        foreach (var item in JsonApiResponse.ToList())
        {
            var emotionSectionModel = GetPersonDetailsModel(item);
            personDetailsModelList.Add(emotionSectionModel);
        }
    }
    return PartialView("~/Views/Image/_PersonDetailsSection.cshtml", personDetailsModelList);
}
catch (Exception ex)
{
    var errorPersonDetailsModel = new PersonDetailsModel();
    errorPersonDetailsModel.Error = ex.ToString();
    personDetailsModelList.Add(errorPersonDetailsModel);
    return PartialView("~/Views/Image/_PersonDetailsSection.cshtml", errorPersonDetailsModel);
}
}

```

The **PersonDetailsModel(JToken json)** method receives a json containing only the face details of a person detected in the image. The method parses the json and returns a Person Details Model containing the following properties: accessories, age, beard, eye makeup, gender, glasses, bald, hair color, lip makeup, moustache, sideburns, smile.

```

private PersonDetailsModel GetPersonDetailsModel(JToken json)
{
    var personDetailsModel = new PersonDetailsModel();
    var faceAttributes = json[Constants.FaceAttributes];

    personDetailsModel.Accessories = faceAttributes[Constants.Accessories].ToObject<List<AccessoriesModel>>();
    personDetailsModel.Age = faceAttributes[Constants.Age].ToString();
    personDetailsModel.Beard = faceAttributes[Constants.FacialHair][Constants.Beard].ToString();
    personDetailsModel.EyeMakeup = faceAttributes[Constants.Makeup][Constants.EyeMakeup].ToString();
    personDetailsModel.Gender = faceAttributes[Constants.Gender].ToString();
    personDetailsModel.Glasses = faceAttributes[Constants.Glasses].ToString();
    personDetailsModel.Bald = faceAttributes[Constants.Hair][Constants.Bald].ToString();
    personDetailsModel.HairColor =
faceAttributes[Constants.Hair][Constants.HairColor].ToObject<List<HairColorModel>>();
    personDetailsModel.LipMakeup = faceAttributes[Constants.Makeup][Constants.LipMakeup].ToString();
    personDetailsModel.Moustache = faceAttributes[Constants.FacialHair][Constants.Moustache].ToString();
    personDetailsModel.Sideburns = faceAttributes[Constants.FacialHair][Constants.Sideburns].ToString();
    personDetailsModel.Smile = faceAttributes[Constants.Smile].ToString();
    return personDetailsModel;
}

```

The **AnalyzeImage(HttpPostedFileBase uploadedImage)** method receives an HttpPostedFileBase file corresponding to the uploaded image or to the webcam capture image. The method gets the byte array from the InputStream and calls the asynchronous method

MakeAnalysisRequest(byteArray) which performs the API analyze request and then returns a list of EmotionSectionModel. The EmotionSectionModel list is passed to the partial view “/_EmotionSection.cshtml” where the image analysis results are displayed.

If an error occurs the stack trace is returned on the catch block.

This method is called through the AJAX post to the URL “"/Image/AnalyzeImageFromUrl".

```
[HttpPost]
public async Task<ActionResult> AnalyzeImage(HttpPostedFileBase uploadedImage)
{
    try
    {
        byte[] fileData = null;
        using (var binaryReader = new BinaryReader(uploadedImage.InputStream))
        {
            fileData = binaryReader.ReadBytes(uploadedImage.ContentLength);
        }
        var emotions = await MakeAnalysisRequest(fileData);
        return PartialView("~/Views/Image/_EmotionSection.cshtml", emotions);
    }
    catch (Exception ex)
    {
        var emotionModelList = new List<EmotionSectionModel>();
        emotionModelList.Add(new EmotionSectionModel(ex.ToString()));
        return PartialView("~/Views/Image/_EmotionSection.cshtml", emotionModelList);
    }
}
```

The **AnalyzeImageFromUrl(string imageUrlAddress)** method receives a string representing the image web URL. The method gets the byte array by calling the method GetImageAsByteArrayFromUrl(imageUrlAddress) and then calls the asynchronous method MakeAnalysisRequest(byteArray) which performs the API analyze request and then returns a list of EmotionSectionModel. The EmotionSectionModel list is passed to the partial view “/_EmotionSection.cshtml” where the image analysis results are displayed.

If an error occurs the stack trace is returned on the catch block.

This method is called through the AJAX post to the URL “"/Image/AnalyzeImage".

```
[HttpPost]
public async Task<ActionResult> AnalyzeImageFromUrl(string imageUrlAddress)
{
    try
    {
        byte[] fileData = GetImageAsByteArrayFromUrl(imageUrlAddress);
        var emotions = await MakeAnalysisRequest(fileData);
        return PartialView("~/Views/Image/_EmotionSection.cshtml", emotions);
    }
    catch (Exception ex)
    {
        var emotionModelList = new List<EmotionSectionModel>();
        emotionModelList.Add(new EmotionSectionModel(ex.ToString()));
    }
}
```

```

        return PartialView("~/Views/Image/_EmotionSection.cshtml", emotionModelList);
    }
}

```

The **GetImageAsByteArrayFromUrl(string imagePath)** method receives a string representing the image web URL. The method downloads and returns the byte array through a WebClient. This method is called by the AnalyzeImageFromUrl method.

```

public static byte[] GetImageAsByteArrayFromUrl(string imagePath)
{
    var webClient = new WebClient();
    byte[] imageBytes = webClient.DownloadData(imageFilePath);
    return imageBytes;
}

```

The **MakeAnalysisRequest(byte[] imageData)** method receives a byte array as parameter image to be analyzed. The method gets the analysis of the specified image file by using the Computer Vision REST API.

The API request is created as follows:

- The subscription key is added to the request headers:

```
client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
    Constants.SubscriptionKey);
```
- Assemble the URI for the REST API Call by appending the request parameters to the URI base:

```
string uri = Constants.UriBase + "?" + requestParameters;
```
- The content type header is set:

```
content.Headers.ContentType = new
    MediaTypeHeaderValue("application/octet-stream");
```
- The REST API call is executed:

```
response = await client.PostAsync(uri, content);
```
- The Json response is parsed and saved in the private field "JsonApiResponse" for later operations (such as getting more face details or get face rectangles)

```
JArray jsonArrayObj = JArray.Parse(contentString);
    JsonApiResponse = jsonArrayObj;
```
- Finally, the method GetEmotionsFromJsonArray is called and its result is returned

```

static async Task<List<EmotionSectionModel>> MakeAnalysisRequest(byte[] byteData)
{
    HttpClient client = new HttpClient();

    // Request headers.
    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", Constants.SubscriptionKey);

    // Request parameters. A third optional parameter is "details".
    string requestParameters =
"returnFaceId=true&returnFaceLandmarks=false&returnFaceAttributes=age,gender,headPose,smile,facialHair,glasses,emotion,hair,makeup,occlusion,accessories,blur,exposure,noise";

    // Assemble the URI for the REST API Call.
    string uri = Constants.UriBase + "?" + requestParameters;

    HttpResponseMessage response;

    // Request body. Posts a locally stored JPEG image.
    /byte[] byteData = GetImageAsByteArray(imageFilePath);

    using (ByteArrayContent content = new ByteArrayContent(byteData))
    {
        // This example uses content type "application/octet-stream".
        // The other content types you can use are "application/json" and "multipart/form-data".
        content.Headers.ContentType = new MediaTypeHeaderValue("application/octet-stream");

        // Execute the REST API call.
        response = await client.PostAsync(uri, content);

        // Get the JSON response.
        string contentString = await response.Content.ReadAsStringAsync();

        // JSON response.
        var jsonString = JsonPrettyPrint(contentString);
        JArray jsonArrayObj = JArray.Parse(contentString);
        JsonApiResponse = jsonArrayObj;

        var emotions = GetEmotionsFromJsonArray(jsonArrayObj);
        return emotions;
    }
}

```

The **GetImageAsByteArray(string imageFilePath)** method receives as parameter a string: `imageFilePath` representing the image file to read. The method returns the contents of the specified file as a byte array.

```

static byte[] GetImageAsByteArray(string imageFilePath)
{
    FileStream fileStream = new FileStream(imageFilePath, FileMode.Open, FileAccess.Read);
    BinaryReader binaryReader = new BinaryReader(fileStream);
    return binaryReader.ReadBytes((int)fileStream.Length);
}

```

The **GetEmotionsFromJsonArray(JArray jsonArray)** method retrieves a Json array as parameter containing the details for each person detected in the image. The method parses the Json Array and depending on the number of the detected persons it creates and returns a list of “EmotionSectionModel”s. If the number of detected persons is 0 then an error message is set to be

displayed later “No person detected, please try to use a better image!”). If the number of persons is 1 or greater for each person an “EmotionSectionModel” is created by calling the “GetEmotionsModel” method and added to the emotionSectionModelList; the main emotion is also detected by calling the “ComputeMainEmotion” method.

```
static List<EmotionSectionModel> GetEmotionsFromJsonArray(JArray jsonArray)
{
    var emotionSectionModelList = new List<EmotionSectionModel>();
    if (jsonArray.Count() == 0)
    {
        var emotionSectionModel = new EmotionSectionModel("No person detected, please try to use a better image!");
        emotionSectionModelList.Add(emotionSectionModel);
        return emotionSectionModelList;
    }

    if (jsonArray.Count() == 1)
    {
        var emotionSectionModel = GetEmotionsModel(jsonArray.First());
        emotionSectionModel.MainEmotion = "The main emotion seems to be: " +
            ComputeMainEmotion(jsonArray.First()[Constants.FaceAttributes][Constants.Emotion]);
        emotionSectionModelList.Add(emotionSectionModel);
        return emotionSectionModelList;
    }

    if (jsonArray.Count() > 1)
    {
        int count = 1;
        foreach (var item in jsonArray.ToList())
        {
            var emotionSectionModel = GetEmotionsModel(item);
            var mainEmotion = ComputeMainEmotion(item[Constants.FaceAttributes][Constants.Emotion]);
            emotionSectionModel.MainEmotion += "Person number " + count++ + " seems to mainly feel " +
                mainEmotion;
            emotionSectionModelList.Add(emotionSectionModel);
        }
    }

    return emotionSectionModelList;
}
```

The **GetEmotionsModel(JToken json)** method receives as parameter a json, it retrieves the face detected emotions and their confidence scores and then it creates and returns an EmotionSectionModel populated with each emotion (anger, contempt, disgust, fear, happiness, neutral, sadness, surprise) and its confidence score.

```
private static EmotionSectionModel GetEmotionsModel(JToken json)
{
    var emotionSectionModel = new EmotionSectionModel();
    var emotionJson = json[Constants.FaceAttributes][Constants.Emotion];
    emotionSectionModel.Anger = emotionJson[Constants.Anger].ToString();
    emotionSectionModel.Contempt = emotionJson[Constants.Contempt].ToString();
    emotionSectionModel.Disgust = emotionJson[Constants.Disgust].ToString();
    emotionSectionModel.Fear = emotionJson[Constants.Fear].ToString();
    emotionSectionModel.Happiness = emotionJson[Constants.Happiness].ToString();
    emotionSectionModel.Neutral = emotionJson[Constants.Neutral].ToString();
    emotionSectionModel.Sadness = emotionJson[Constants.Sadness].ToString();
    emotionSectionModel.Surprise = emotionJson[Constants.Surprise].ToString();
}
```

```

    return emotionSectionModel;
}

```

The **ComputeMainEmotion(JToken item)** method receives a json containing only the face emotions and their confidence scores of a person detected in the image. The method creates a key - value dictionary, the key being represented by the emotion name and the value being represented by the confidence of the emotion. Finally, the method returns the emotion with the highest confidence from the dictionary, as the main detected emotion.

```

private static string ComputeMainEmotion(JToken item)
{
    Dictionary<string, double> emotionsDictionary = new Dictionary<string, double>
    {
        { Constants.Anger, (double)item[Constants.Anger] },
        { Constants.Contempt, (double)item[Constants.Contempt] },
        { Constants.Disgust, (double)item[Constants.Disgust] },
        { Constants.Fear, (double)item[Constants.Fear] },
        { Constants.Happiness, (double)item[Constants.Happiness] },
        { Constants.Neutral, (double)item[Constants.Neutral] },
        { Constants.Sadness, (double)item[Constants.Sadness] },
        { Constants.Surprise, (double)item[Constants.Surprise] }
    };

    return emotionsDictionary.FirstOrDefault(pair => pair.Value == emotionsDictionary.Values.Max()).Key;
}

```

The Image controller is responsible to perform the analysis requests to the Face API and using the JSON response (error or success) it creates the corresponding models for the partial views that will be displayed.

The **partial views and models** used in the method mentioned above are:

Emotion Section partial view “_EmotionSection.cshtml”:

```

@model List<EmotionalRecognition.Models.EmotionSectionModel>

@section Scripts
{
    <script src="@Url.Content("~/Scripts/Home/Index.js")"
    type="text/javascript"></script>
}

<h1>Image analysis results</h1>

<div class="emotionSectionDiv">
    @foreach (var item in @Model)
    {
        if (!string.IsNullOrEmpty(@item.Error))
        {
            <div class="card personEmotionCard">
                <span style="max-width: 490px">@item.Error</span>
            </div>
        }
        else

```



```

{
    <div class="card personEmotionCard">
        @if (Model.IndexOf(item) == 0)
        {
            <span>Detection result:</span>
            if (Model.Count > 1)
            {
                <span>@Model.Count faces detected</span>
            }
            else
            {
                <span>@Model.Count face detected</span>
            }
        }
        <span>@item.MainEmotion</span>
        <span>{</span>
        <table>
            <tr class="emotionsGrid">
                <td>Anger: @item.Anger</td>
                <td>Contempt: @item.Contempt</td>
                <td>Disgust: @item.Disgust</td>
                <td>Fear: @item.Fear</td>
                <td>Happiness: @item.Happiness</td>
                <td>Neutral: @item.Neutral</td>
                <td>Sadness: @item.Sadness</td>
                <td>Surprise: @item.Surprise</td>
            </tr>
        </table>
        <span>}</span>
    </div>
}
}
</div>
@if (Model.FirstOrDefault(emotion => emotion.Error != null) == null)
{
    <button id="getMoreInfoAboutPersonsBtn" onclick="getMoreInfoAboutPersons();
return false;" class="btn btn-primary btn-lg upload-button" style="margin-
bottom: 5px;">Get More Info &raquo;</button>
}

```

EmotionSectionModel:

```

namespace EmotionalRecognition.Models
{
    public class EmotionSectionModel
    {
        public EmotionSectionModel (string error)
        {
            Error = error;
        }

        public EmotionSectionModel()
        {
        }

        public string MainEmotion { get; set; }
        public string Anger { get; set; }
        public string Contempt { get; set; }
    }
}

```

```

        public string Disgust { get; set; }
        public string Fear { get; set; }
        public string Happiness { get; set; }
        public string Neutral { get; set; }
        public string Sadness { get; set; }
        public string Surprise { get; set; }
        public string Error { get; set; }
    }
}

```

Person Details Section partial view “_PersonDetailsSection.cshtml”:

```
@model List<EmotionalRecognition.Models.PersonDetailsModel>
```

```
@section Scripts
```

```
{
    <script src="@Url.Content("~/Scripts/Home/Index.js")"
    type="text/javascript"></script>
}
```

```
<h2 style="margin-top: 20px; margin-bottom: 10px">More details results</h2>
```

```
<div class="emotionSectionDiv">
```

```
    @foreach (var item in @Model)
    {
```

```
        if (!string.IsNullOrEmpty(@item.Error))
```

```
        {
            <span>@item.Error</span>
        }
```

```
    else
```

```
    {
        <div class="card personEmotionCard">
```

```
            @if (Model.Count == 1)
            {
```

```
                <span>Person Details</span>
            }
```

```
        else
```

```
        {
            <span>Person number @(@Model.IndexOf(item) + 1) details</span>
        }
```

```
    <span>{</span>
    <table>
```

```
        <tr class="emotionsGrid">
```

```
            <td>Gender: @item.Gender</td>
```

```
            <td>Age: @item.Age</td>
```

```
            <td>Smile: @item.Smile</td>
```

```
            <td>Glasses: @item.Glasses</td>
```

```
            <td>Face Hair</td>
```

```
            <td>{</td>
```

```
                <td class="personDetailsGridRows">Moustache:
```

```
@item.Moustache</td>
```

```
                <td class="personDetailsGridRows">Beard: @item.Beard</td>
```

```
                <td class="personDetailsGridRows">Sideburns:
```

```
@item.Sideburns</td>
```

```
            <td>}</td>
```

```
            <td>Makeup</td>
```

```
        <td>{</td>
```

```

        <td class="personDetailsGridRows">Eye Makeup:
@item.EyeMakeup</td>
        <td class="personDetailsGridRows">Lip Makeup:
@item.LipMakeup</td>
    <td></td>
    @if (item.Accessories.Count > 0) {
        <td>Accessories</td>
        <td></td>
        foreach (var acc in @item.Accessories)
        {
            <td class="personDetailsGridRows">Type: @acc.Type -
Confidence: @acc.Confidence</td>
        }
        <td></td>
    }
    <td>Bald: @item.Bald</td>
    @if (item.HairColor.Count > 0) {
        <td>Hair Colors</td>
        <td></td>
        foreach (var hc in @item.HairColor)
        {
            <td class="personDetailsGridRows">Color: @hc.Color -
Confidence: @hc.Confidence</td>
        }
        <td></td>
    }
    </tr>
</table>
<span></span>
</div>
    }
}
</div>

```

Person Details Model:

```

namespace EmotionalRecognition.Models
{
    public class PersonDetailsModel
    {
        public string Smile { get; set; }
        public string Gender { get; set; }
        public string Age { get; set; }
        public string Moustache { get; set; }
        public string Beard { get; set; }
        public string Sideburns { get; set; }
        public string Glasses { get; set; }
        public string EyeMakeup { get; set; }
        public string LipMakeup { get; set; }
        public List<AccessoriesModel> Accessories { get; set; }
        public string Bald { get; set; }
        public List<HairColorModel> HairColor { get; set; }
        public string Error { get; set; }
    }
}

```

Accessories Model

```
namespace EmotionalRecognition.Models
{
    public class AccessoriesModel
    {
        public string Type { get; set; }
        public double Confidence { get; set; }
    }
}
```

FacePosition Model

```
namespace EmotionalRecognition.Models
{
    public class FacePosition
    {
        public int Top { get; set; }
        public int Left { get; set; }
        public int Width { get; set; }
        public int Height { get; set; }
    }
}
```

HairColorModel

```
namespace EmotionalRecognition.Models
{
    public class HairColorModel
    {
        public string Color { get; set; }
        public double Confidence { get; set; }
    }
}
```

5 LICENSE

In March 2012, Scott Guthrie announced on his blog that Microsoft had released part of its web stack (including ASP.NET MVC, Razor and Web API) under an open source license (Apache License 2.0)

The project is under the protection of Apache License 2.0. This is a permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

For the Cognitive Services from Microsoft (Face API) a license is also needed. For this project development I used more trial API Keys. The API Key is added to the Request Headers in order to provide access for using the API functionality.

6 BIBLIOGRAPHY

[FREEMAN14] *Pro ASP.NET MVC 5 Platform* by Adam Freeman, 2014

[SKEET08] *C# in Depth* by Jon Skeet, 2008

[CHOWDHURY17] *Mastering Visual Studio 2017* by Kunal Chowdhury, 2017

[WOODS11] *APIs: A Strategy Guide* by Dan Woods, Daniel Jacobson, and Greg Brail, 2011

[RICHARDSON13] *RESTful Web APIs: Services for a Changing World* by Leonard Richardson, 2013

[SPURLOCK13] *Bootstrap: Responsive Web Development* by Jake Spurlock, 2013

[Pilgrim10] *HTML5: Up and Running* by Mark Pilgrim, 2010

[MEYER00] *Cascading Style Sheets: The Definitive Guide* by Eric A. Meyer, 2000

[CROCKFORD08] *JavaScript: The Good Parts* by Douglas Crockford, 2008

[FLANAGAN10] *JQuery Pocket Reference* by David Flanagan, 2010

[CHACON09] *Pro Git* by Scott Chacon, 2009

7 WEB REFERENCES

[ASP.NETMVC] – ASP.NET MVC, available online at
https://en.wikipedia.org/wiki/ASP.NET_MVC

[ASP] – ASP and ASP.NET Tutorials, available online at
<https://www.w3schools.com/asp/>

[MVC] – ASP.NET MVC Tutorial, available online at
http://www-db.deis.unibo.it/courses/TW/DOCS/w3schools/aspnet/mvc_intro.asp.html

[MVC] – Learn About ASP.NET MVC, available online at
<https://www.asp.net/mvc>

[TDD] – Test-driven development, available online at
https://en.wikipedia.org/wiki/Test-driven_development

[ASP.NETMVC] – Getting Started with ASP.NET MVC 5, available online at
<https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/getting-started>

[VS] – Visual Studio, available online at
<https://visualstudio.microsoft.com>

[AZURE] – Microsoft Azure, available online at
https://en.wikipedia.org/wiki/Microsoft_Azure

[C#] – C# Tutorial available online at
<https://www.tutorialspoint.com/csharp/index.htm>

[C#01] – C# and its Features, 2001, available online at
<https://www.c-sharpcorner.com/article/C-Sharp-and-its-features/>

[API17] – What is Emotion API?, 2017, available online at
<https://docs.microsoft.com/en-us/azure/cognitive-services/emotion/home>

[API] – Emotion API, available online at
<https://westus.dev.cognitive.microsoft.com/docs/services/5639d931ca73072154c1ce89/operations/563b31ea778daf121cc3a5fa>

[API16] – API types, 2016, available online at
<https://ffeathers.wordpress.com/2014/02/16/api-types/>

[Bootstrap] – Bootstrap, available online at
<https://getbootstrap.com/>

[BOOTSTRAP] – Bootstrap (front-end framework), available online at [https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))

[BUNDLE13] – What is a bundle?, 2013, available online at <http://vswebessentials.com/features/bundling>

[BOOTSTRAP] – 10 Best Reasons to Use Bootstrap for Amazing Web Designs, 2015, available online at <https://www.devsaran.com/blog/10-best-reasons-use-bootstrap-amazing-web-designs>

[HTML] – HTML, available online at <https://en.wikipedia.org/wiki/HTML>

[HTML] – HTML Element, available online at https://en.wikipedia.org/wiki/HTML_element

[DOM] – Document Object Model, available online at https://en.wikipedia.org/wiki/Document_Object_Model

[CSS] – Cascading Style Sheets, available online at https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[LESS] – Less (stylesheet language), available online at [https://en.wikipedia.org/wiki/Less_\(stylesheet_language\)](https://en.wikipedia.org/wiki/Less_(stylesheet_language))

[SASS] – Sass (stylesheet language), available online at [https://en.wikipedia.org/wiki/Sass_\(stylesheet_language\)](https://en.wikipedia.org/wiki/Sass_(stylesheet_language))

[JS] – JavaScript basics, available online at https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

[JQUERY] – jQuery - Overview, available online at <https://www.tutorialspoint.com/jquery/jquery-overview.htm>

[VC] – Version control, available online at https://en.wikipedia.org/wiki/Version_control

[GIT] – GIT, available online at <https://git-scm.com/>

[GIT] – GIT, available online at <https://en.wikipedia.org/wiki/Git>

[GITHUB] – GITHUB, available online at <https://en.wikipedia.org/wiki/GitHub>

8 CD / DVD



9 INDEX

B

Bibliography, 86

C

CD / DVD, 89

Content, 18

I

Introduction, 16

L

License, 85

T

Table of figures, xiv

W

Web References, 87