HELP University
university of achievers

Affective Movie Evaluator

BIT304 FINAL YEAR PROJECT I

B1301746 Ibrahim Mohamed Shaatha
B1301511 Elebe Faith Alfred

Submitted to the

FACULTY OF COMPUTING AND DIGITAL TECHNOLOGY
(SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY)

In partial fulfilment of the requirements
for the degree of

HUBIT
HUBSC2

HELP UNIVERSITY
APRIL 2019

Final Report I: Affective Movie Evaluator

Ibrahim M. Shaatha, Faith Alfred

April 15, 2019

Contents

List of Tables

List of Figures

Abstract

For a project to be successfully implemented, it requires meet-ups with stakeholders, team members and any external member that will contribute positively to the success of the project. For our project we didnt have a need to meet with external stakeholders, but it was necessary for us to meet with each other (project team member), and our supervisor. These meetings have enabled us to manage our project well, share ideas and limit our work to a specific audience. It has also helped us to eliminate some work that we think is necessary but later realised it would have just been a waste of time and resources.

Declarations

Student Names:

Mohamed Shaatha Ibrahim (B1301746)

Elebe Faith Alfred (B1301511)

Date: 12-04-2019

Acknowledgement

Aknowledgements here.

Part I

Proposal

Chapter 1: Introduction

Movie production studios use test screening in order to determine whether their movies will succeed. Recently, internet movie distributors such as Netflix and Amazon have started taking more control of the production of their content, it is more beneficial for studios to re-edit their movie once they can predict that the product is going to fail. While there are automated sentiment analysis tools used to measure how well the audience reacts to test screening, most of the feedback is still taken from questionnaires and the audiences subjective evaluation.

We aim to provide an automated, movie evaluation system which is going to analyse the audience watching the movie and predict an overall score. Our system is intended to supplement the existing manual process of test screenings. We are not proposing a system to replace the existing subjective evaluations given by the audience.

The aim of this project is to develop a system that can evaluate how good a movie is, by analysing both face and body postures of the audience. In order for this to work we provide the system a video file consisting of audience reactions, and the system will tell us how good a movie is within a rating metrics, we will define later. It is important for us to state, that we are not proposing a real-time system here; the system accepts a recording of the raw data - which is currently defined as video file, and can be any other form of measurement - as the primary input.

A system like this can aid us develop further, a more general system that can evaluate how good a certain product or presentation is by analysing the audience reaction limited by a certain context. In our case the product is a movie, and how the audience reacts may depend on what genre of movie they are watching, or what kind of experience they were expecting beforehand. It is likely that our system may not work well without knowing these information. Systems like these are also related to computer vision with Emotion AI technologies, such as face based lie detectors, which uses the same techniques to work.

Chapter 2: Background

For a long time movie industry have used statistical modeling and machine learning to predict, the success of movies based on high level data. While there have been several works in early to mid 2000s, which takes in multimedia information to form a representation for the users emotions (Zhang, Tian, Jiang, Huang, & Gao, 2008; Wu et al., 2008), the earliest work we could find that specifically focus on movie scenes was by Soleyman et. al.

Quite recently, there have been interest in using deep learning techniques to analyse audiences facial data such as (Saha, Navarathna, Helminger, & Weber, 2018), to predict whether an audience enjoyed the movie, most notably research work that has been funded by The Walt Disney Company (Deng et al., 2017). The purpose of their research is to use facial emotion recognition techniques, to analyse the emotion of the audience in order to improve the movies.

Research has shown that over 90% of our communication can be non verbal (Algorithmia, 2018). Non- verbal communication which includes facial expressions and body postures among others can be used to determine the emotional state of a human being. According to Paul Ekman (Ekman P, 1992), there are basically six types of human emotions namely: happiness, sadness, disgust, fear, surprise, and anger.

The system we are proposing is going to take in a video file of an audience reacting to a movie and attempts to determine whether the audience liked that movie. It will use emotion recognition and body pose to quantify the overall reaction in order to determine whether the viewer enjoyed the movie or not.

Although, idea of such a system is not new, we have not been able to find an existing system that considers both body pose and facial emotion data to create a model that can predict. Previous work such as (V Ramanarayanan et al., 2015), which evaluates a presenter by using a very similar technique that we are planning, performed very poorly compared to the human judges. However, we are a bit more optimistic, since movies are designed to elicit certain emotional responses and audiences in average exhibit similar reactions. Such a system, maybe useful in the film industry to analyse the reactions of the test audience as a supplement to the usual questionnaires. The problem we are trying to solve is that film studios screens movies to test audience, and audience writes back a subjective feedback usually on paper. Our software is intended to complement the existing evaluation procedure, by providing an automated evaluation of audience's experience that is less subjective and provide a numerical score that represents audience's sentiment more accurately.

Affective Analysis is already used by marketing giants such as Coca-Cola to predict the effectiveness of their advertisement on target test audiences. Affective movie analysis could prove to be valuable to streaming services that distributes their own content and have more control over the productions such as NetFlix.

Chapter 3: Issues

Our system is not real-time but still requires heavy computing power in order to run. This means the admin will often have to run multiple processes for analysis and come back later for the result. Our system does not support multiple processes yet, while it is possible for our system to run a process on a remote server or a cloud computer service via ssh and http, the admin still have to wait for the analysis to be completed.

In order to minimize the scale of the problem, the first few iterations of the problem deals with only a single audience member. The largest audience we will be designing our system will be for a group of two to four person, under controlled lighting environment. Unless we have access to infrared cameras and better computing facilites, we will not be able to overcome these limitations.

Our video can only have one person in front of the camera. The application is not able to give proper result if there are faces in the background when the audience member is reacting. Presence of another person in the room currently effects the resulting data, and it is very time consuming for a human to verify whether the recorded data does not have any background face. However, as we wrote our detection algorithm to choose the largest region of interest in the image, the fail cases we have observed usually consists of background posters and wallpapers with larger face than our subject.

One of the weakness we have is in the accuracy of the face detection algorithm. OpenCV's default facial detection algorithm, (Viola-Jones full frontal model) requires the audience member to face the camera directly. However, we plan to implement a more accurate face detection algorithm at the risk of increasing computing time.

Right now our system uses an externel media player known as VLC media player, to play the movie while being recorded in use case 1. The system admin is usually required to install the software manually. However on Windows based systems, it is sometimes not possible to find the

Chapter 4: Benefits and Constraints

For our current iteration the system can analyse only one audience member. The reason for this limitation is that we initially wanted to start the project with a smaller scope and also to reduce the amount of tasks we might have for the first iteration. Even in our future iterations, we will not be analysing and audience consisting of more than a handful of person, unless we get better camera equipments.

Another issue faced by us is that facial emotion recogntion is not working that well, for now. In order to improve the accuracy of the FER model we plan to write separate test cases.

Since our system takes a long time to analyse, we had to modify our "Analyse Video" use case into two seperate parts. Instead of directly providing a video file to run analysis and immediately getting the results and visualisations, our system actually requires the admin to run two seperate instance of analysis, facial emotion analysis (FER) and pose analysis (BEGR). Running an instance of analysis will produce the required data file, which can then be visualised as a time-series and processed to be sent for our movie evaluation ML model.

Chapter 5: Project Aims and Objectives

The aim of our project is to develop a system that can evaluate how good a movie is, by analysing both face and body postures of the audience. In order for this to work we provide the system a video file consisting of audience reactions, and the system will tell us how good a movie is within a rating metrics, we will define later. It is important for us to state, that we are not proposing a real-time system here; the system accepts a recording of the raw data - which is currently defined as video file, and can be any other form of measurement - as the primary input.

- system like this is very useful for movie studios, which often relies on test audience in order to predict whether their movies are going to be successful and a system like this can aid in measuring the audience reaction more accurately.

The objectives we have set for our project are the following: Define the metrics by which the system is going to give the score before we create the evaluation model. (e.g: value between 0.0-1.0 or a discrete grading system, like A+, A,

- Research the requirements of the dataset, and methodologies before week 3.
- Find or create the required test datasets for in order to complete objective 6.
  - Dataset for the emotion recognition subsystem
  - Dataset for body pose subsystem.
  - Dataset for the whole system: Video recordings of audiences reacting to movies.
  - Determine accuracy of the overall project by comparing its output to existing scoring systems.
- To prove that we are using state of the art technologies in our system; produce:
  - Accuracy report for Emotion recognition
  - Accuracy report Body pose/action estimation

Chapter 6: Project Scope

- For developement purpose, we will be dealing with short movies mostly.
- We will be recording atleast 12 subjects, watching multiple movies.

Before we proceed, we would like to borrow a very recent terminology from Andrej Karpathy that has been in use in the industry. *Software 2.0* refers to any software that is written without direct human involvement, with the help of another software. We chose to use this term in document to highlight the fact that the developement approach and the documentation practices used for the machine learning portions of the code is slightly different than the rest, and it will not be honest from our part to describe the process as the same. Though term originated in an article he wrote, we have seen the term used within a very few academic papers (Ratner, Hancock, Dunnmon, Goldman, & Ré, 2018). In contrast, *Software 1.0* refers to any type of software that a human is directly responsible for creating and we will be only using the term to distinguish it, if necessary.

A large portion of the project is in Software 2.0, therefore we do realize we have to spend a lot of time collecting data, analysing the data and evaluating our application. For this reason our primary methodology is described as SEMMA methodology, even though for iteration 1 we actually use a mixed method.

Chapter 7: Software and Hardware Requirements

7.1 Funding Sources

As mentioned before we do not have any corporate and industry backing, therefore we are expecting HELP School of ICT to provide any possible funding we may require, if requested. Since the University already have most of the assets we require for our FYP1, we will require permission from the IT department to use the facilities, for our project. As for FYP2, the tools and resources we require for data collection and obtaining ground truth can only be determined after we complete the first part of our project.

7.2 Minimum hardware requirements

Although, our system can run with a laptop with a webcam we designed the system with the following requirements in mind.

Table 7.1: Minimum Hardware Requirements

No	Item	Usage	Qty
1	VGA USB Camera	Used to record audience	1
2	Computer Terminal (Desktop/Laptop /SmallFactor)	Our application need to run on an computer with a display terminal.	1

7.3 Recommended hardware requirements

Table 7.2: Recommended Hardware Requirements

No	Item	Usage	Qty
1	HD Camera	Used to record audience	1
2	High-end Desktop Computer	Our application need to run on an computer with a display terminal.	1
3	CUDA compatible Graphics Card - NVDIA geforce gtx 1080 ti	Used to improve video analysis models	1

Chapter 8: Developement Methodology

To give you the context of the reasoning behind our decision, we would like to discuss a brief overview of commonly used development methodologies first. The most commonly followed methodologies for data-science projects are either SEMMA or CRISP-DR methodologies. SEMMA is a methodology developed by SAS institute, while CRISPR-DR method was originally developed in a joint effort by five companies, and further improved by data mining industry, most notably by IBM.

The main difference between both methodologies are that CRISPR-DR considers business understanding of the process and also considers model deployment and the results. SEMMA is mainly used for data mining projects, which does not consider business aspect. The methodology we chose for our project is SEMMA methodology.

The reason we chose SEMMA is that, we wanted to keep option generalise our problem scope later, for example we wanted to apply what the system does in other scenarios than movies. Also, business project timelines and research project timelines are usually incompatible, and this is a fact we need to consider while planning early on, since machine learning projects falls under the latter. So not considering the business aspects of the project, will be an advantage for us given the time we have.

SEMMA methodology consists of 5 phases (Sample, Explore, Modify, Model and Assess). SEMMA methodology gives us freedom to reiterate from any step, as we require. It does not have a definitive cycle for iteration, which makes it a more simple and less rigid framework to follow, compared to CRISP-DR (Palacios, 2017). Sample: First step of of SEMMA involves collecting all data samples. Explore: Understand the problem through exploration of sample data. Exploration can be helped through visualisations and other data analytic tools. Modify: Next steps, involves selecting which variables/features can be used, how they can be transformed or processed in order to create a model. Model phase consists of selecting and creating a model Assess: Finally we evaluate how well the model performed, and determine whether it is useful in real world situation. In order to properly use SEMMA methodology into our project we are going to we are going to collectively apply the methodology to three separate components of our system that we have previously identified; the facial emotion subsystem, body emotion subsystem and movie evaluation model. After we started planning, we realized given only two team members, it was actually impossible for us to go through the whole system with one iteration within one semester.

Therefore, we have split the project into two phases. Each phase is considered iteration by itself, because during the first phase, we build the first two components of our pipeline simultaneously (by two members), such that they both go through the SEMMA process at the same time. During the process, we also ensure both components are integrated before we start the next phase.

Then, we will start next iteration; go back to the sample stage to work on the final part of our project. For example, after the Assess phase of the emotion models we will be starting again from Sample phase again, and ensure everything in the pipeline is working well. This does not mean we define the term phase and iteration as being interchangeable.

Eventually, given the results it is possible we may plan a new iteration, if necessary. For example, if after assessing the Movie Evaluation Model, it performed poorly, we may go back to either Modify stage to pick better features to be passed into the model or we can even go back to Model stage to pick a different model.

The first planned iteration of our project we will be focused on building the facial emotion and body pose emotion subsystem, which will be for simplicity hence referred to as Emotion Model and Pose Model, respectively. The second part of our project will be the combined system along with Movie Evaluation Model. Most of our work on Movie Evaluation Model, is currently planned to commence on the second part of our internship (FYP2), and the gantt chart we have proposed in the Appendix section might significantly change.

As our primary version control system we will be using gitlab as our github repository. Also, since SEMMA does not define a project management style, we will be using kanban/scrum like board feature on gitlab, to organize the tasks that we will be doing. However, we will not be adhering to any strict project management style, which is very prevalent in the software industry.

As for the specific techniques we will use throughout the development, our idea is to analyse existing literatures, and compare notes and follow general data mining/modeling techniques with fewer brainstorm sessions. In order to obtain the ground truth while collecting the data, our current idea is to use questionnaires after recording a subjects reaction. Alternatively, its possible for us to measure blood pressure or any other means of observation if possible, and we will consider that for our FYP2.

Part II

Project Management Plan

Chapter 1: Introduction

The aim of our project is to develop a system that can evaluate how good a movie is, by analysing both face and body postures of the audience. In order for this to work we provide the system a video file consisting of audience reactions, and the system will tell us how good a movie is within a rating metrics, we will define later. It is important for us to state, that we are not proposing a real-time system here; the system accepts a recording of the raw data - which is currently defined as video file, and can be any other form of measurement - as the primary input. A system like this is very useful for movie studios, which often relies on test audience in order to predict whether their movies are going to be successful and a system like this can aid in measuring the audience reaction more accurately.

1.1 OBJECTIVES OF THE PROJECT

The objectives we have set for our project are the following: Define the metrics by which the system is going to give the score before we create the evaluation model. (e.g: value between 0.0-1.0 or a discrete grading system, like A+, A, B) Research the requirements of the dataset, and methodologies before week 3. Find or create the required test datasets for in order to complete objective 6. Dataset for the emotion recognition subsystem Dataset for body pose subsystem. Dataset for the whole system: Video recordings of audiences reacting to movies. Determine accuracy of the overall project by comparing its output to existing scoring systems. To prove that we are using state of the art technologies in our system; produce: Accuracy report for Emotion recognition Accuracy report Body pose/action estimation

1.2 OUTPUTS EXPECTED FROM THE PROJECT

The project will be divided into two phases; initially, we will be working on two seperate system that capture the emotion and pose respectively, into meaningful data and then streamline our work into a single codebase. The second phase of our project will be working together to create the movie evaluation model, which takes in the processed data from our previous system and gives us a prediction of how well the audience liked that movie.

The main focus of our project will be developing the Movie Evaluation Model that we have discussed earlier, in our project proposal. However, since the movie evaluation model we have proposed requires all the emotion data, we need to use a working emotion recognition system first. We may use any python/tensorflow based off the shelf or open source system for that purpose.

The first part of our project is choosing and testing the preliminary subsystems of the pipeline. Note that since we use off the shelf code for the first phase of the project, there is no training dataset required, we expect that pre-trained models are provided by the library we choose. Also, in order to properly validate the system it is ideal to avoid using the original training dataset that was used to create the pre-trained model as our test datasets.

The deliverables consists of:

- A Facial emotion recognition subsystem. (Faith)
  - Data collection: Facial dataset and emotion dataset.
  - Evaluate facial detection library
  - Evaluate emotion recognition library
- Test script
  - Performance Report.
- Body pose emotion (Body language) subsystem. (Ibrahim)
  - Data Collection: Body pose data
  - Test script
  - Performance Report.
- Integrated codebase: Contains code to load and run the previous two modules on a frame of a video file, and prepare data to feed the movie evaluation model.
- Movie Evaluation model.
- Collect movie dataset: Record audience reacting to movies.
- Report

Chapter 2: Work Breakdown Structure

During our analysis phase we broke down the project into largest completable tasks for each phase. In-order to identify the completable tasks, we thought of the project in terms of user-story/use-case perspective and also by identifying the independent components of the system, from our analysis. We thought of the first input data that will pass through the pipeline, and how it will be processed along to get the results we wanted. For example, if our initial data is video files captured by a camera device, the system must have to at somepoint interact with a camera object, therefore it is very likely during our design phase that we may need to create a camera object for a module.

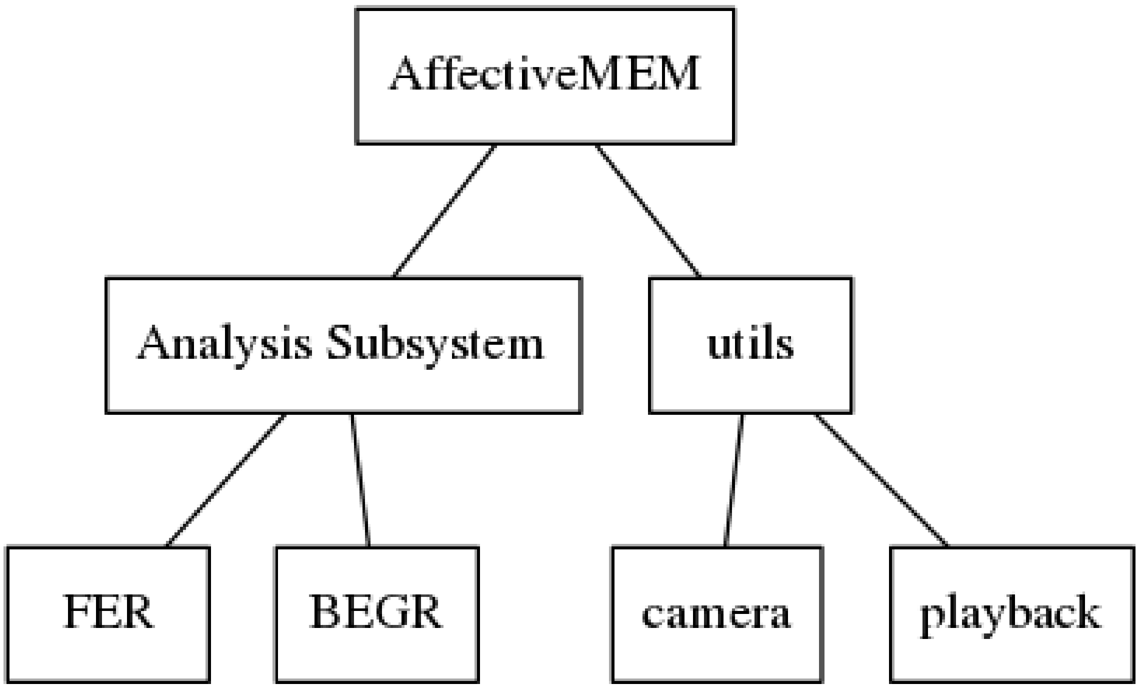


Figure 2.1: Task in terms of complexity

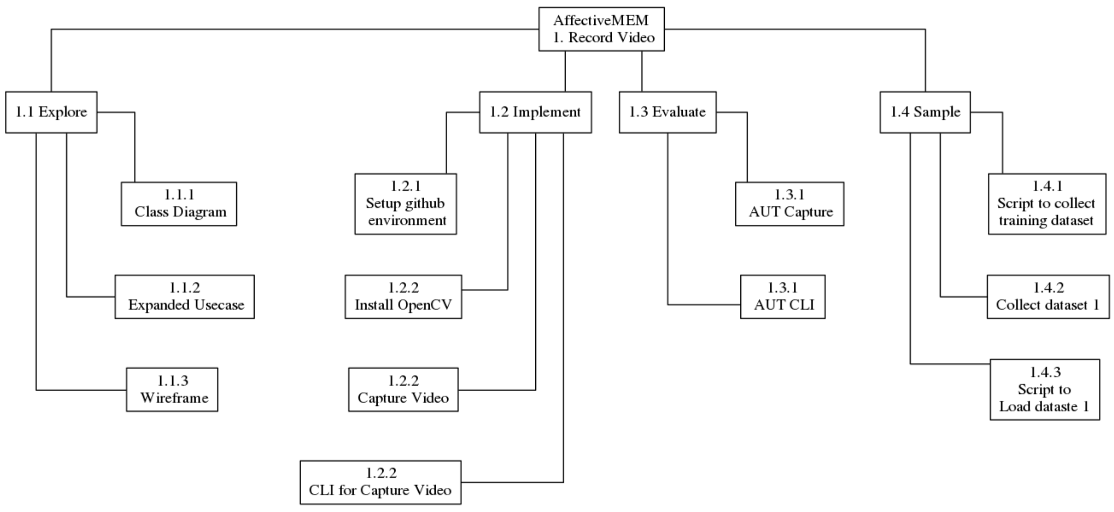


Table 2.1: Work breakdown tasks

Phase	Task	Start	Finish	Size
1	1. Class Diagram	12/03/19	16/03/19	M
	2. Expanded Use-case	12/03/19	16/03/19	S
	3. Wireframe	12/03/19	16/03/19	S
	1. Setup github environment			S
	2. Install OpenCV			S
	3. Capture Video in a folder			L
	4. CLI to Capture Video			M
	1. AUT Capture			M
	2. AUT CLI			S
	1. Script to collect training dataset			M
	2. Collect dataset 1			L
	3. Script to load dataset1			M

Every task on the list have a deliverable.

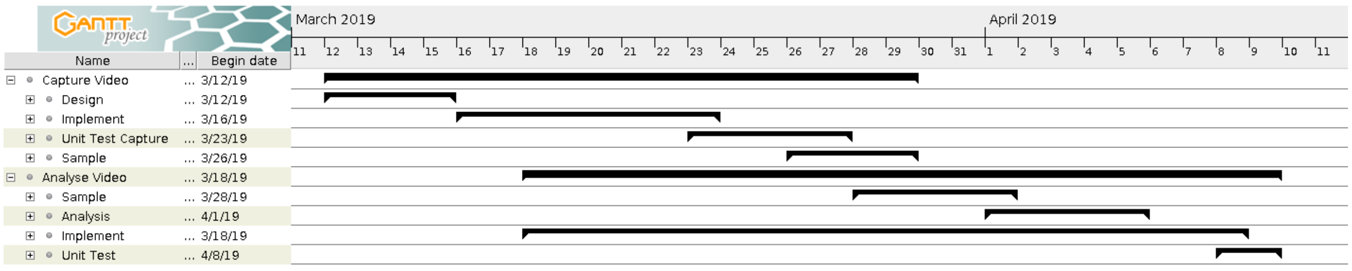
Chapter 3: Risk Management Plan

Table 3.1: Risk Management Plan

#	**Description**	**Probability**	**Impact**	**Mitigation Strategy**
1	Cannot collect dataset to test movie evaluation system.	40%	4	Use online reaction videos for shorter movie clips.
2	Insuffficent memory to train the model	20%	4	Ask the department for cloud computer access or more physical computer resources
3	Evaluation model is not performing as expected, because features extracted are not sufficent to represent properly	5%	2	Use another mean to take the measurements, use a thermal camera or heartbeat measuring instrument. If possible use new measurements to create a baseline to compare the evaluation model.

Chapter 4: Project timeline

You can find our baseline gantt chart for estimated project plan in Appendix A. We created the work breakdown structure during our project plan, and we grouped each task under a phase in our methodology. Then we estimated time for each task and prioritsed them, finally creating a project timeline in a gantt chart format.



Name	...	Begin date
Capture Video	...	3/12/19
Design	...	3/12/19
Implement	...	3/16/19
Unit Test Capture	...	3/23/19
Sample	...	3/26/19
Analyse Video	...	3/18/19
Sample	...	3/28/19
Analysis	...	4/1/19
Implement	...	3/18/19
Unit Test	...	4/8/19

Figure 4.1: Overview of baseline gantt chart

Part III

Requirement Analysis

Chapter 1: Introduction

Since the conception of the project plan, our initial goal for the first iteration was to create a system that is going to analyse the facial emotion and body posture emotion independently and construct a data representation such as a time-series from the analysis. We delegated the primary goal of our project, which is to use that data to train and create a model that can score movies for later iterations intentionally, so it will give us time to reconsider our methods and also create dataset in between the downtime.

Like any other software developement project, our approach was to first of all analyse the system and identify. Practically, most of the requirement analysis was done while we were creating the project plan, while we did not thoroughly document it in the proposal itself. This part of the document was later updated to reflect the change in number of use-cases in our iteration 1, after we realised that we are not able to fit the three initial use-cases we planned into the first iteration.

Along with the project's functional requirements and non-functional requirements, we will also try to give an overview of the requirements from the machine learning perspective. We will try to state the ML problem, and define the requirements and structure for the problem. It should be noted by now, our system will consist multiple ML algorithms interacting with each other and a portion of has to be designed by us.

We also wanted to complete all the software design and engineering intensive tasks as early as possible so we can focus on the machine learning aspects of our project. Thus, a large portion of this iteration is writing the code for user interaction, and designing processes within the system that can perform independent of the type of user interface.

Initially, we like we planned we worked on the facial analysis, and posture analysis portion of the code seperately. Later on we refactored and integrated the code as seperate modules into a single codebase with the help of object-oriented programming.

Chapter 2: Requirement Summary

The first thing we did during our first meeting was infact to identify all the stakeholders and actors who interacted with the system. While we discussed the naming conventions of the actors involved during the later meetings, it was early on established that only one user directly interacts with the system and can be considered to be the primary user.

Our systems consists of mainly this one user controlling the application directly. Ideally this user is a system operator, hired by the production company with a knowledgeable background in IT and system administration. Throughout this document we refer to this user, simply as *Admin*. We should note that the user *Admin* is not really a recognized object within the system and thus does not have a login function. *Admin* refers to whoever is operating the system.

2.1 Actors and Stakeholders

2.1.1 Admin

An *Admin* can use the system to record audiences watching a movie. Right now, in order to do that, the movie must exist within the computer system first. Admin is supposed to launch our application, and provide the details of the audience member and the movie, because the system does not yet store the metadata of the movies yet. The movie object is not considered as an object of the system in this iteration.

- Admin is able to Record a video of a *Subject* watching a movie.
- Admin is able to Analyse a video of a *subject*
    - Produce time-series.

2.1.2 Subject

Subject is not considered a primary actor in the system. The only interaction between the subject and the system occurs when the admin records the subject watching a movie in the first use-case. The admin initiates the use-case and subject passively sits infronts of the camera, until the movie is completed, whole while the admin actually have the control of the system such that they can terminate the recording process or wait until the movie is finished.

While the UML for iteration 1 might not directly hint that our system considers *Subject* as a seperate entity for design purpose, it is going to be added as an Object in our future iteration. We did not include the *Subject* as an object that is in the system, yet.

2.2 Object represented in the system

For this iteration, all use-cases are going to be involved with only two object. Object Sample represents a sample recording of an audience video, while the object movie represents a Movie that an audience is watching. It is important that these two logical objects are well defined within the system, as we might later need to categorize and display the statistics for future use cases.

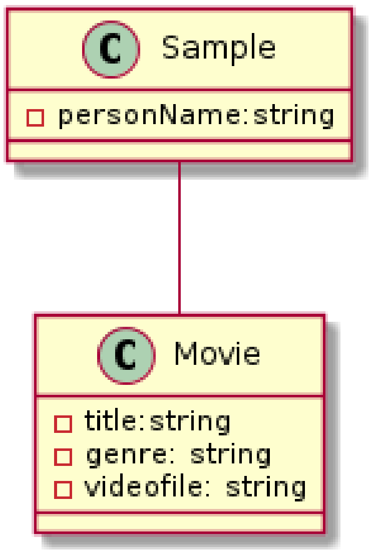


Figure 2.1: Objects in the system

Chapter 3: Functional Requirements

3.1 Use-Case Diagrams

While designing of the use-cases we only thought from a logical/business perspective and not included the "technical" usecases, as our methodology and approach does not heavily rely on UML.

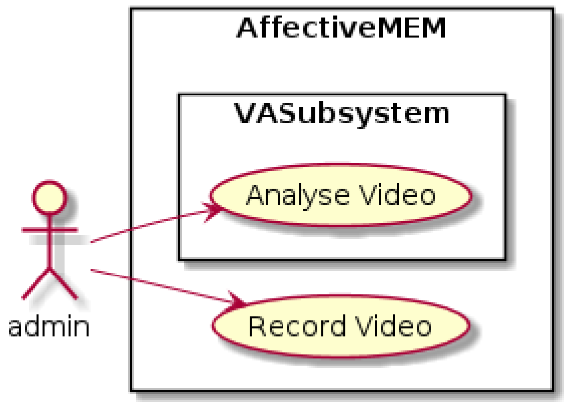
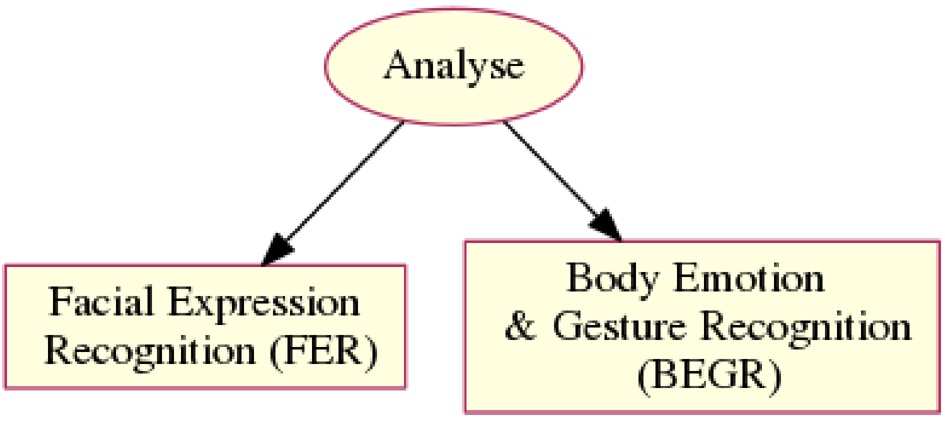


Figure 3.1: Iteration 1: Before integration

In figure 3.2, you can see that the Analyse Video is bound within a subsystem of Affective Movie Evaluator. The reason

While there is a weak implementation of other use cases, the two use-cases shown in the diagram 3.2, are the main goal of our current iteration until now.



Analyse
Facial Expression Recognition (FER)
Body Emotion & Gesture Recognition (BEGR)

Figure 3.2: Analyse uscase with subsystems

It would have been ideal for the sake of this documentation for Analyse use case to include two "technical" use-cases, "Analyse FER" and "Analyse BEGR", so that it is easier to credit the two team members, but it would not have been accurate. Ideally a use case should show an added value to a system, and seperating them does not really contribute the value to from either logical or design perspective.

3.2 High-Level Use Cases

Table 3.1: Analysis usecases

Use Case 1	Record Video [1]
Goal in Context	To record an audience member's video and store it.
Primary Actors Secondary Actor	- Admin - Audience
Description	Use camera to record and store video session of an audience watching a movie screening.

Use Case 2	Analyse Video [2]
Goal in Context	To extract an audience's facial and body pose data from a video file and store it as readable data.
Primary Actors Secondary Actor	- Admin - Audience
Description	Use camera to record and store video session of an audience watching a movie screening.

[1]Use case was designed by faith and implemented by Ibrahim.

[2]FER portion of use-case was created by Faith and BEGR was created by Ibrahim.

Chapter 4: Non-Functional Requirements

4.1 Technical Requirements

4.1.1 Prototype 1 - BIT304

We recommend to use a moderately powerful desktop computer to run the application. For video capture, it currently uses a webcam, though the application is currently compatible with any other type of live video capture stream. Ideally, a dual monitor system must be used one facing the audience and one facing the *Admin*, who will be interacting with the control panel. The system was actually tested on a laptop computer and we assume that it is also possible to setup and run this application on most standard laptops without any issue.

Almost all of our application is written using Python 3. We used the Anaconda distribution of python and it's environment and package management system because it reduced the compatibility issues associated with installing packages such as tensorflow across different systems.

Table 4.1: Minimum Software Requirements

Software
Windows 7/8

1. Recommended Setup

    - Linux based system, preferrably Debian variant
    - System Utilites
        - git
        - conda
    - Intel based processor preferrably greater than 5th Generation i5
    - 8GB RAM

4.2 Usability Requirements

Our standard user documentation is written as a README file in markdown format. Since the project is currently available in GitHub, the documentation is the first thing the user can see of our project. The purpose of the user documentation is to describe the whole project briefly, and guide the user to install and setup the application.

The user is able to interact with the application through commandline and GUI.

4.3 Reliability Requirements

In our current iteration, all the files are stored in the filesystem within the project file. Our version control system (git) does not track the data files and models, since they are considered large and we do not think it will be effective to use VCS to track data files. Therefore, datasets and results must be shared and versioned manually, we will have to back it up separately and share it on internet with email and file sharing services.

One of the scripts we wrote that will run when a new user is going to setup the system, will actually download the pre-trained models used by subsystems from the internet. This seems to be a standard practice as even larger projects such as OpenPose often does not store their models in the github repository.

4.4 Security Requirements

By design, our system does not have an authentication system nor it is considered required. We consider that if deployed, most production studios have their own security systems to control access.

The only time the application uses network currently is to access the internet and download the required models and software.

One concern a production studio may have is regarding the security of test audience data that is stored. Depending on the contract they have with their own test audience, the test audience may be concerned with their privacy if the data is leaked or shared without their permission for other projects.

4.4.1 FYP2

We plan to implement the cloud analysis use-case for iteration 3.

Part IV

Iterations

Chapter 1: Iteration Plans

Early on we planned two iteration for FYP 1, but as mentioned in the requirement analysis we had to change that because of time constraint and risks of not completing the objectives of our FYP 1.

As shown in our project timeline, we combined SEMMA phases with SDLC phases in our project timeline. The SEMMA phase are not analogous to an SDLC phase, and we've learned that there may exist no standard for such analogy. For example most software engineers and feature engineers tend to write code during the Sample phase while ML engineers may work on something else. For later iterations, we are going to follow the SEMMA phases more rigidly as we will be working more on the machine learning aspects of the project.

1.1 Current Iteration

We only have one Iteration for this part of our final year project. Initially the work was divided between the two part. Use case one was primarily developed by Ibrahim and designed by Faith. Use-case two's main two sub-components were developed and tested independently and later on we re-integrated our work into a single codebase.

Table 1.1: Iteration Plan

FYP	**Iteration**	**Use Cases**	**Proposed Start Date**	**Proposed End Date**
1	1	1,2	11/03/19	09/04/19
2	2,3	2,3,4		

1.2 Future Iterations

[3]Future use cases are described under the conclusion chapter.

Table 1.2: Future Iterations

Iteration	**Use Cases** [1]
2	3, 4
3	4, 5

Chapter 2: Iteration 1

2.1 Introduction

Like we mentioned before this iteration implements only two use-cases, but it does not mean the amount of tasks we have is smaller than iteration 2, which we have planned that involves more use-cases. Our plan for this iteration was changed throughout the developement, project. Initially we had 2 iterations and considered adding more use-cases that manages movies and samples within the system. However once we finished our work breakdown and started to sort out our tasks, we figured out it will be a risk considering all the tasks we have and we might not be able to complete our goals within timespan.

So iteration 1 was expanded to cover most of this semester and we focused on two use-cases from the user-perspective. From the system perspective, we started working on the two main analysis subsystems once our on use-case two commenced.

2.2 Purpose

The purpose of this iteration is mainly to make sure our application can record a video and extract the primary data, required for machine learning from the video. This means the two analysis subsystems must be designed tested, and they should be integrated into the system.

By the end of this iteration the system should be able to do the following:

1. **Capture Video**: The admin should be able to record video by using the command line interface and graphical user interface, to create a video file.

2. **Analyse Video**: The admin must be able to create time-series data from an existing recording of a subject.

2.3 Context

On a grand scale, by the end of this iteration we will have a codebase and an fixed project structure. It should be easier to navigate and design for our future iterations, since we will be more familiar with how we are going to work and what exactly needs to be documented. Most importantly, the review of the first iteration is actually going to help us estimate and prioritize the tasks for future iterations because it is going to show how much we overestimate or under-estimate the size of a task.

2.4 Schedule of Iteration Workflow

Table 2.1: Schedule of Iterations Workflow

Workflow	**Start**	**End**	**Duration**
1.Capture Video	11/03/19	30/03/19	19 days
Design	11/03/19	16/03/19	
Implement	16/03/19	24/03/19	
Evaluate	22/03/19	28/03/19	
Sample	26/03/19	30/03/19	
2. Analyse Video	18/03/19	09/04/19	22 days
Sample	29/03/19	02/04/19	
Explore/Design	01/04/19	06/04/19	
Implement	18/03/19	09/04/19	
Modify	08/04/19	09/04/19	
Test	08/04/19	09/04/19	

2.5 Iteration Schedule Breakdown

Table 2.2: Schedule of Tasks

Use case	**Task Name**	**Start**	**Finish**	**Est Time**
1	**1.1 Design (Explore)** 1.1.1 Class Diagram 1.1.2 Expanded Use-Case 1.1.3 Wireframes	11/03/2019	16/03/2019	5
1	**1.2 Implement**	16/03/19	24/03/19	8

Continued on next page

Continued from previous page

Use case	Task Name	Start	Finish	Est Time
	1.2.1 Setup github environment 1.2.2 Install OpenCV 1.2.3 Capture video in a folder 1.2.4 CLI to capture video			
1	**1.3 Evaluate** 1.3.1 AUT capture 1.3.2 AUT CLI	22/03/19	28/03/19	6
1	**1.4 Sample** 1.4.1 Script to collect training dataset 1.4.2 Collect Test Dataset 1 1.4.3 Write script to load test dataset	26/03/19	30/03/19	4
2	**2.1 Sample** 2.1.1 Collect Test Dataset 2 2.1.1 Download FER dataset 2.1.2 Script to load FER dataset	29/03/19	02/04/19	4
2	**2.2 Explore/Design** 2.2.1 Split and label Test Dataset 2 2.2.2 Pipeline Diagram 2.2.3 Interaction Diagram (SSD) 2.2.4 Design Class Diagram 2.2.5 Wireframe	01/04/19	06/04/19	5
2	**Implement** 2.3.1 Setup FER library 2.3.2 Setup OpenPose Library 2.3.3 Generate FER time series 2.3.4 Generate Pose time series 2.3.5 Integrate FER analysis to UI 2.3.6 Integrate BEGR analysis to UI 2.3.7 Analyse Video CLI	18/03/19	09/04/19	22
2	**2.4 Modify** 2.4.1 Define combined time-series file format	07/04/19	09/04/19	2

Continued on next page

Continued from previous page

Use case	Task Name	Start	Finish	Est Time
	2.4.2 Code to Visualise Time Series 2.4.3 Refactor code			
2	**2.5 Evaluate** 2.5.1 Write automated unit test 2.5.2 FER Accuracy test 2.5.3 Pose accuracy test	07/04/19	09/04/19	2

2.6 Resource Summary

Resources used for this project

Table 2.3: Hardware Resources used for iteration

No	Hardware	Usage	Qty
1	Laptops with webcams	Testing and Developement	2
2	HD Webcam	Capture sample and testing	1
3	Microsoft Azure Cloud VM	To speed up analysis for testing purpose	1

Software Resources used for this project

Table 2.4: Software Resources used for iteration

No	Software	Usage	Qty
1	Python 3 (conda distribution)	Programming language and runtime environment	1
2	Tensorflow	A lower level machine learning API for python which interacts with hardware, used to train, load and predict ML/DNN models.	1

Continued on next page

Continued from previous page

No	Software	Usage	Qty
3	Keras	A higher level python library which can be used as an abstraction over tensorflow, or other lower level API. Like tensorflow optimized for deep deep learning and general ML tasks, but more simple to use.	1
4	Click	Python libary used to create the command line interface	1
5	wxPython	Python implementation of the popular wx widgets toolkit. Used to create our GUI	1
6	pytube	Enables python script to interact with youtube. We use this in data collection scrips and also by scripts to download the test case movies.	1
7	OpenCV python	Python API for OpenCV library. OpenCV is the standard toolkit used to manipulate images and computer vision tasks. We use it for variety of purpose including reading from webcam and performing image manipulations.	1

2.7 Evaluation Criterea

The criteria for evaluation of the whole system is divided into three parts. We intend to test the software

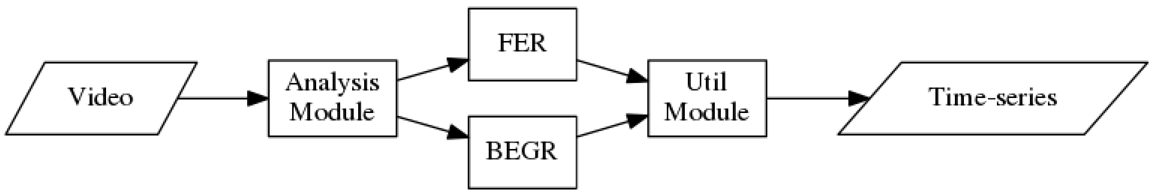
1. FER Evaluation
2. BEGR Evaluation

The approach to test the *Software 1.0* parts and the *Software 2.0* (machine learning) parts of the system are very different. The software engineering aspects and machine learning parts of the code must be tested using very serpe

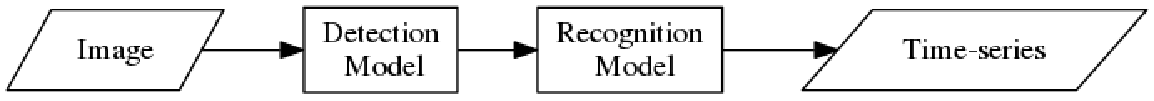
In this iteration we do not evaluate the speed and accuracy performance of the entire pipeline, from the beginning, something we intend to do in all the future iterations.

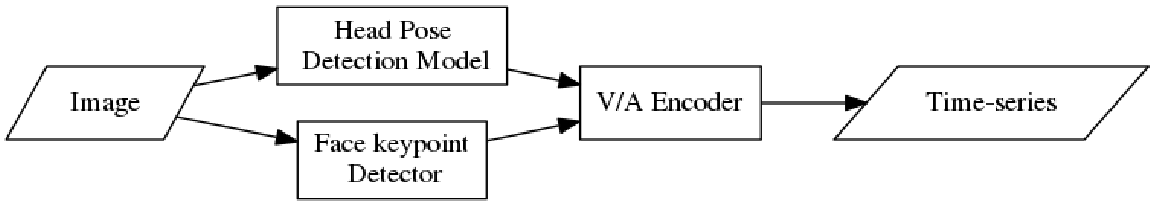
2.8 Analysis and Design Artefacts

2.8.1 Pipeline diagrams



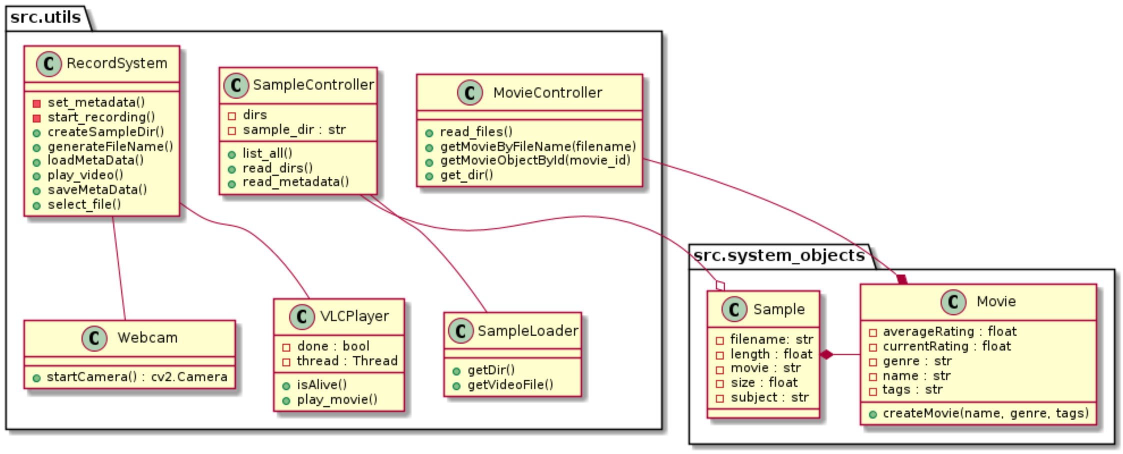
Even though it is possible to run *FER* and *BEGR* simultaneous and in parallel by design, during the implementation phase we realised our development systems are not capable of running the subsystems in parallel without running out of memory. Therefore, we decided that the video will pass through the analysis pipeline twice to exctract the time-series data.





The valence arousal encoder is currently not built, instead we use a hard coded module as a replacement. Currently all the V/A encoder returns is a 1 dimensional value that we call 'interest'. If the subject is facing the camera the algorithm returns a higher value, otherwise we return a lower value.

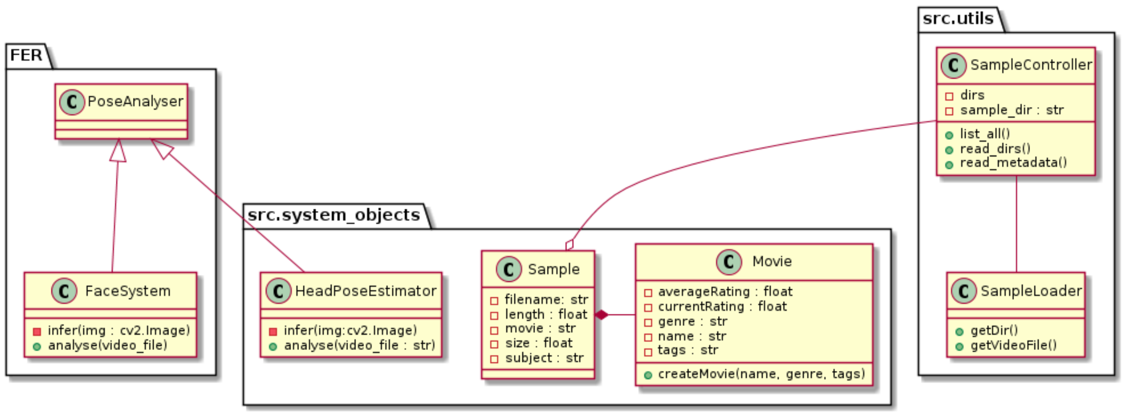
2.8.2 Class Diagrams



The classes RecordSystem, SampleController and MovieController are respectively used by the User interface code for the use case. We do not have a presentation layer abstraction between the actual user interface code and the *Controller* classes.

One design principle we tried to follow loosely is the Single-Responsible principle, as you can see from the diagram. A class should have one responsibility, but we did not adher strictly to it.

Webcam module helps setting up the camera.



Note since the current BEGR subsystem's full body pose recognizer is too slow, we replaced it with another model only measuring head pose and renamed the package *begr* as *head_pose*. Package *head_pose* is just a placeholder for *begr* until we find a better replacement.

Also, the fact that BEGR code is not in a seperate package and FER code is in a separate package, is because of a miscommunication and oversight between the two programmers who worked on this during the developement and we decided not to refactor the mistake until, next iteration due to lack of time. BEGR should be refactored into it's own package by in the future.

Chapter 3: Iteration I: Implementation and Testing

Our manual tests corresponds to the use-cases, for demonstration purpose we isolate the components associated with the use-case. We do this manual process only for the sake of documentation, while most other smaller systems are tested with the help of automated pythons automated unit testing framework.

3.1 Implementation

Our application is mostly written in python with the exception of few OS specific shell scripts used to setup and configure. For developement we used conda distribution of python, along with its package and environment manager. The main reason we chose conda was it's ease of use, when installing certain packages such as tensorflow for various platforms compared to pythons default package manager.

```
~/affective_movie_evaluator/$ conda create -name affmem_env \
                                                  pip tensorflow python=3.5
~/affective_movie_evaluator/$ conda activate affmem_env
~/affective_movie_evaluator/$ conda activate
(affmem_env) ~/affective_movie_evaluator/$
```

The shell command snippet above shows an example scenario of creating a conda environment and activating the environment for the project. Right now we recommend all *Admins* to work using conda environment to avoid common multiple dependency version issues users face when they use the global python environment.

All of the data was stored directly on disk drive, and the system identifies the objects with the help of metadata written in JSON format. Since we are not expecting search functions, that will traverse through the disk drive and most machine learning algorithms either loads datasets in batches or in entirety we do not think a database is necessary. A search with the current system will perform linearly $O(n)$ and an insertion will cost constant time.

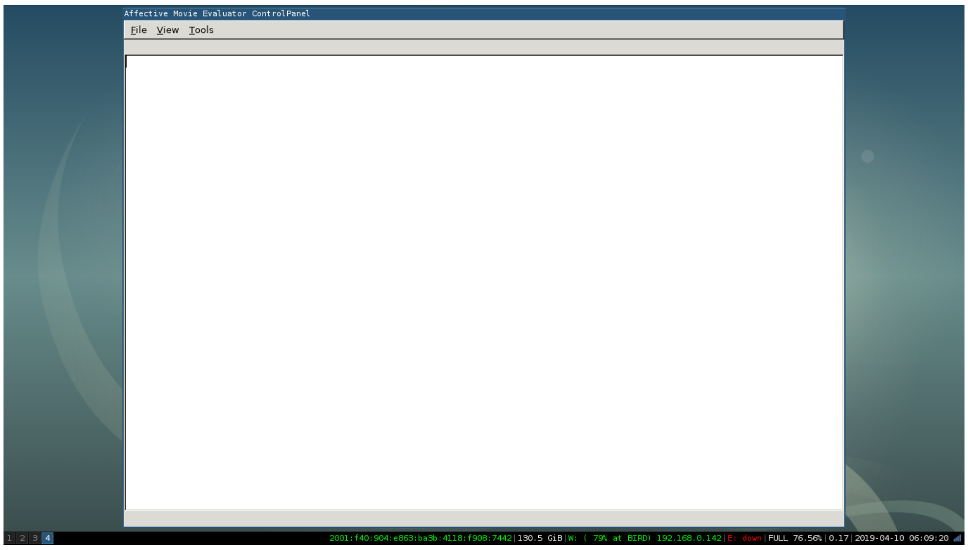
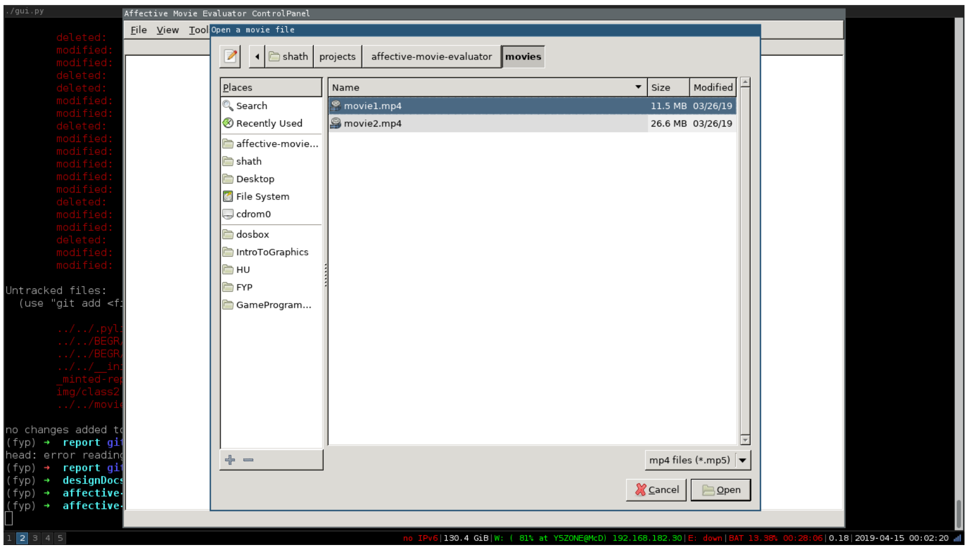

Affective Movie Evaluator ControlPanel
File View Tools
1 2 3 4
2001:f40:904:e863:ba3b:4118:f908:7442|130.5 GiB|W: (79% at BIRD) 192.168.0.142|E: down|FULL 76.56%|0.17|2019-04-10 06:09:20

Figure 3.1: The GUI Application

Figure 3.1 shows the initial screen the admin will see once they start the application. The empty text area shown in the screenshot is going to display error error messages for the admin, and was also used as the console log by us while were developing the application.

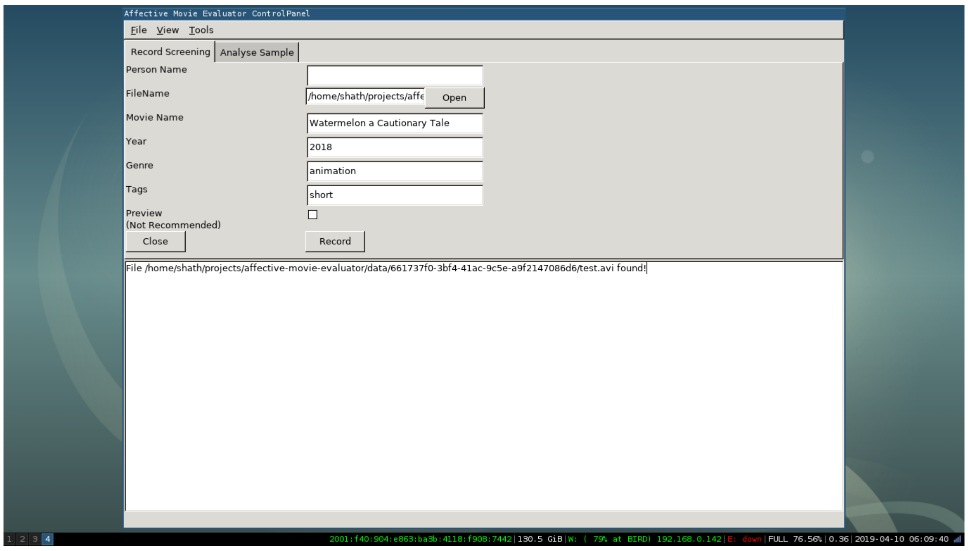

```
./gui.py
        deleted:
        modified:
        modified:
        deleted:
        deleted:
        modified:
        modified:
        deleted:
        modified:
        modified:
        modified:
        modified:
        modified:
        deleted:
        modified:
        modified:
        deleted:
        modified:
        modified:

Untracked files:
  (use "git add <f

        ../../.pyl
        ../../BEGR
        ../../BEGR
        ../../__in
        _minted-re
        img/class2
        ../../movi

no changes added t
(fyp) → report gi
head: error readin
(fyp) → report gi
(fyp) → designDoc
(fyp) → affective
(fyp) → affective
```

Name	Size	Modified
movie1.mp4	11.5 MB	03/26/19
movie2.mp4	26.6 MB	03/26/19

Figure 3.2: Record Sample: 1. Select the movie file

If the *Admin* wants to record a new sample, they will have to select a movie stored from the disk drives. The movie files are stored in a the 'movies' directory as shown in Figure 3.2, along with the metadata file which describes each movie.

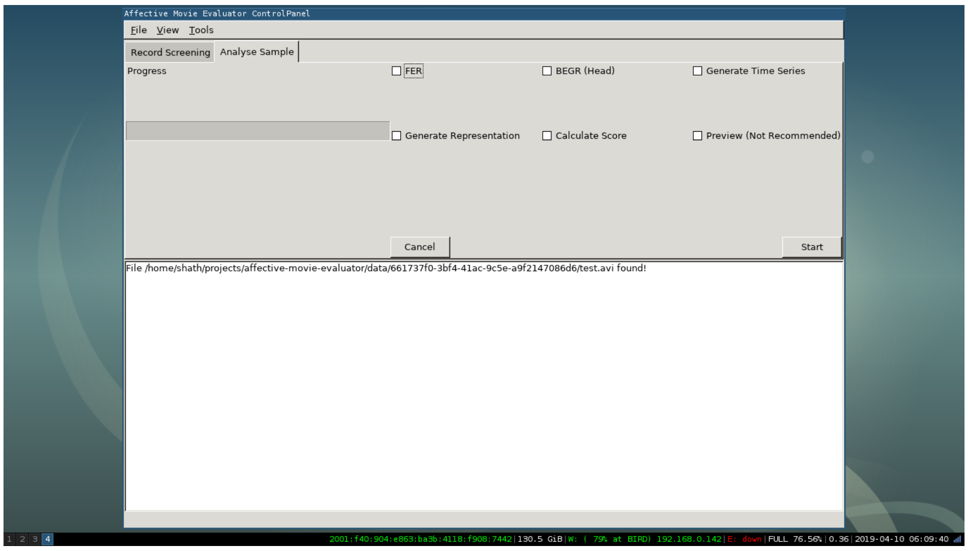
Our overall strategy for implementation can be describe as a test driven bottom up approach. For modules that uses machine learning we have a different approach, instead of writing tests for accuracy or simple integration we write a procedural test file, then convert into Object Oriented code which is compatible with our Analyse interface.



Person Name
FileName /home/shath/projects/affe
Movie Name Watermelon a Cautionary Tale
Year 2018
Genre animation
Tags short
Preview (Not Recommended)

```
File /home/shath/projects/affective-movie-evaluator/data/661737f0-3bf4-41ac-9c5e-a9f2147086d6/test.avi found!
```

Figure 3.3: Record Sample: 2. Fill in the blanks



```
File /home/shath/projects/affective-movie-evaluator/data/661737f0-3bf4-41ac-9c5e-a9f2147086d6/test.avi found!
```

Figure 3.4: Analyse Screen

3.2 Testing

Table 3.1: Table of Strategy

No.	Strategy
1	Create the smallest possible class
2	Write the unit test for the class and improve the class until it passes all the basic test conditions
3	Write the user interface code

Table 3.2: Table of Strategy for Models

No.	Strategy
1	Write driver code for used for testing purpose
2	Write the class which can import the model and perform predictions
3	Keep on improving the class until the realtime tests pass

We did not use the main dataset and movie diretory for testing purpose, but instead created mock data for that purpose. For example, in order to test the SampleLoader which loads the Sample for other module given the parameter, we actually created a test directory without actual samples but metadata files.

Table 3.3: Test Plan

Type of testing	Approach	Subject
Automated Unit Testing Integration Testing	White Black Box	Unit: RecordSystem Usecase 1, Usecase 2

All automated unit tests are prefixed with the letter A to distinguish it from manual tests, which are prefixed with the letter M. Manual unit testing requires user to run and observe the results for each test case, and tests may require user input unlike automated unit tests.

Table 3.4: Test Plan

Type of testing	**Approach**	**Use Case**	**Task**
Automated Unit Testing	White	1. Record Video	

3.3 Unit Testing

Our goal of unit testing is to test the smallest possible software components within the system seperately and manually without where it is possible. For more internal components that mainly interacts with other components, we do no rigidly test all possible fail cases as some fail cases may be triggered by a programming mistakes.

Unit testing was implemented under the task, "Test CLI".

Table 3.5: Automated Unit Tests

#	**Module**	**Package**	**Usecase**
A1	SampleLoader	src.util	1,2
A2	SampleController	src.util	1,2

Table 3.6: Manual Unit Tests

#	**Module**	**Package**	**Usecase**
M1	Webcam	src.utils	1
M2	VLCPlayer	src.utils	1

3.3.1 UnitTest: A1

Table 3.7: Unit Test A1

Unit Test Plan	
Module Name:SampleLoader	file: "./test/path_test.py"
1. Module Overview	
The purpose of the SampleLoader is to aid the SampleController load the Sample files from the directory. Inspite of it's name the actual responsibility of the class is to resolve the directory path of of a sample folder and nothing else.	

Continued on next page

Continued from previous page

Unit Test Plan
1.1 Input to Module
Input 'param1' will passed as argument 1 of the constructor. sys = SampleLoader(param1)
1.2 Outputs from Module
For every input. - CASE 1: – assert self._sdir == "./data/test/" - CASE 2: testGetVideoFile: assert self.getVideoFile() == "./data/test/test.avi" - CASE 3: testDirSlash:
1.3 LogicFlow / Source
./src/utils.py : SampleLoader
2. Test Data
- "./data/test/" - "./data/test"
2.1 Positive Test Cases
- CASE 1: testGetDir() - CASE 2: testGetVideoFile() - CASE 3: testDirSlash()
2.2 Negative Test Cases
- None
3. Interface Modules
requires: none
4. Test Tools
- unittest - Run unit only: "python3 -m unittest tests.path_test.TestSampleLoader" Part of the test suite. Please run ./run_test.sh on a Linux system to run all tests.

```
(fyp)  affective-movie-evaluator git:(master)  python3 -m unittest tests.path_test.Test
...
----------------------------------------------------------------------
```

```
Ran 3 tests in 0.000s


OK
```

3.3.2 UnitTest A2

Table 3.8: Unit Test A2

Unit Test Plan
Module Name:SampleController file: "./test/path_test.py"
1. Module Overview
The purpose of the SampleController is to handle Create, Retrieve, Update and Delete operations of the Sample Objects within the system.
1.1 Input to Module
Input 'param1' will passed as argument 1 of the constructor. sys = SampleLoader(param1)
1.2 Outputs from Module
expected_data = ['661737f0-3bf4-41ac-9c5e-a9f2147086d6', '1d989665-80ae-4bff-bf59-b5f7691fb3b9'] For every input. - CASE 1: – assert sys.list_all() == expected_data Expected data corresponds to valid sample folders manually created with valid, metadata.
1.3 LogicFlow / Source
./src/utils.py : SampleLoader
2. Test Data
- "./data/test/"
2.1 Positive Test Cases
- CASE 1: test1() Purpose of test1 is to make sure that the list_all() method retrieves only valid Sample folders.
2.2 Negative Test Cases
- None

Continued on next page

Continued from previous page

Unit Test Plan
3. Interface Modules
requires: none
4. Test Tools
- unittest - Run unit only: "python3 -m unittest tests.path_test.TestSampleController" Part of the test suite. Please run ./run_test.sh on a Linux system to run all tests.

```
(fyp)   affective-movie-evaluator git:(master)  python3 -m unittest tests.path_test.Test
.
----------------------------------------------------------------------
Ran 1 test in 0.000s


OK
```

3.3.3 ManualTest M1

Table 3.9: Unit Test M1

Unit Test Plan	
Module Name:Webcam	file: "./test/path_test.py"
1. Module Overview	
Purpose of the webcam module is to provide a standard camera interface through webcam for other modules.	
1.1 Input to Module	
1.2 Outputs from Module	
For every input. - CASE 1: – assert self.__sdir == "./data/test/" - CASE 2: testGetVideoFile: assert self.getVideoFile() == "./data/test/test.avi"	

Continued on next page

Continued from previous page

Unit Test Plan
- CASE 3: testDirSlash:
1.3 LogicFlow / Source
./src/utils.py : SampleLoader
2. Test Data
- "./data/test/" - "./data/test"
2.1 Positive Test Cases
- CASE 1: testGetDir() - CASE 2: testGetVideoFile() - CASE 3: testDirSlash()
2.2 Negative Test Cases
- None
3. Interface Modules
requires: none
4. Test Tools
- unittest - Run unit only: "python3 -m unittest tests.path_test.TestSampleLoader" Part of the test suite. Please run ./run_test.sh on a Linux system to run all tests.

Part V

Appendix

Chapter 1: Appendix A

AffectiveMEM Apr 15, 2019

HELP http://

Project manager	
Project dates	Mar 12, 2019 - Apr 10, 2019
Completion	99%
Tasks	38
Resources	2

AffectiveMEM Apr 15, 2019

Tasks

2

Name	End date	Begin date
Capture Video	3/29/19	3/12/19
Design	3/15/19	3/12/19
1.1.1 Class Diagram	3/13/19	3/12/19
1.1.3 Wireframe	3/12/19	3/12/19
1.1.2 Expanded Use-Case	3/15/19	3/14/19
Unit Test Capture	3/27/19	3/16/19
Unit Test Capture	3/27/19	3/23/19
Unit Test CLI	3/27/19	3/23/19
Implement	3/25/19	3/16/19
Setup githuInstall OpenCV Unit Test CLI CLI Capture Video Unit Test Capture Setup github b	3/16/19	3/16/19
Install OpenCV	3/18/19	3/18/19
Capture Video	3/25/19	3/18/19
CLI	3/23/19	3/18/19
Sample	3/29/19	3/26/19
Script to record dataset	3/29/19	3/26/19
Collect dataset 1	3/28/19	3/27/19
Script to load dataset	3/28/19	3/28/19
Analyse Video	4/9/19	3/18/19
Sample	4/1/19	3/28/19
Collect dataset 2	4/1/19	3/29/19
Script to load FER dataset	3/29/19	3/28/19
Analysis	4/5/19	4/1/19
SSD	4/5/19	4/1/19
Design Class Diagram	4/5/19	4/1/19
Wirefame	4/2/19	4/1/19
Implement	4/9/19	3/18/19
Setup FER Library	3/20/19	3/18/19

AffectiveMEM Apr 15, 2019

Tasks 3

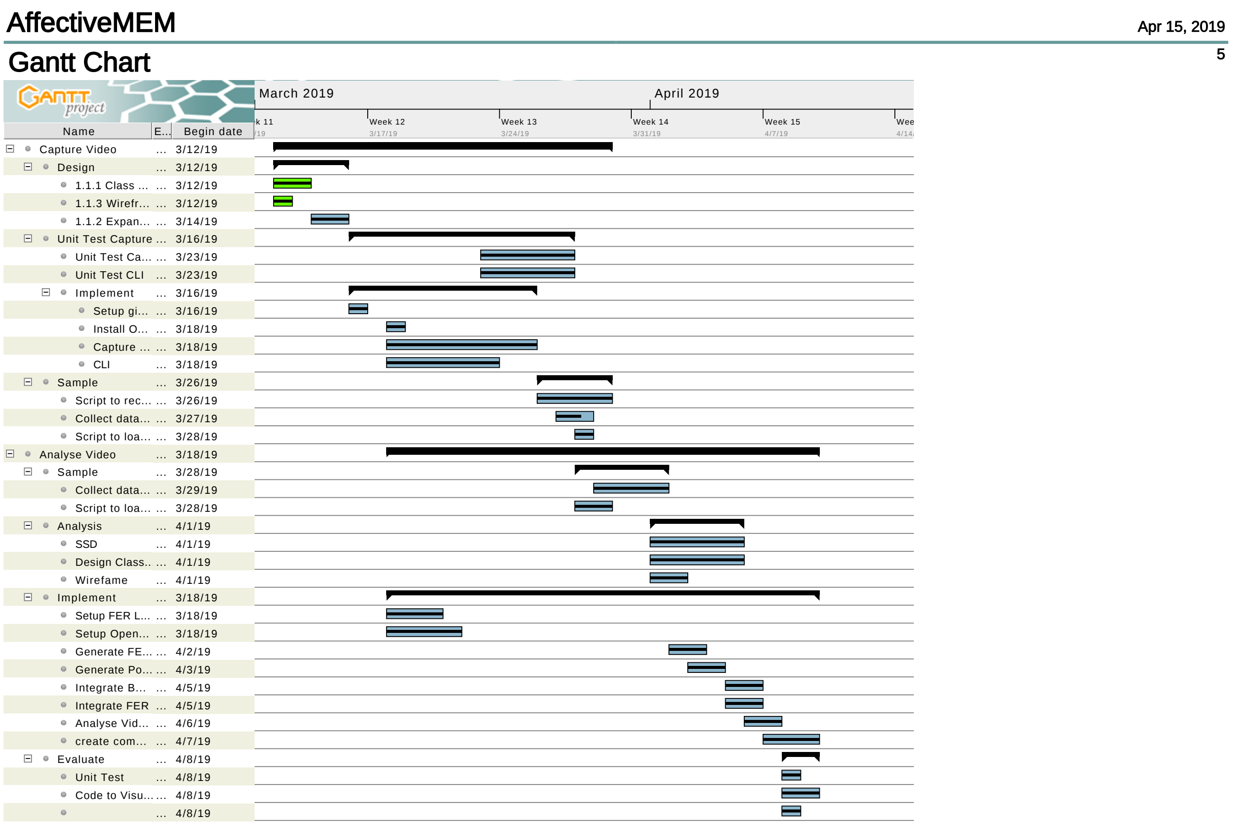
Name	End date	Begin date
Setup OpenPose	3/21/19	3/18/19
Generate FER time-series	4/3/19	4/2/19
Generate Pose TS	4/4/19	4/3/19
Integrate BEGR	4/6/19	4/5/19
Integrate FER	4/6/19	4/5/19
Analyse Video CLI	4/7/19	4/6/19
create combined time-series	4/9/19	4/7/19
Evaluate	4/9/19	4/8/19
Unit Test	4/8/19	4/8/19
Code to Visualize TS	4/9/19	4/8/19
	4/8/19	4/8/19

AffectiveMEM Apr 15, 2019

Resources

4

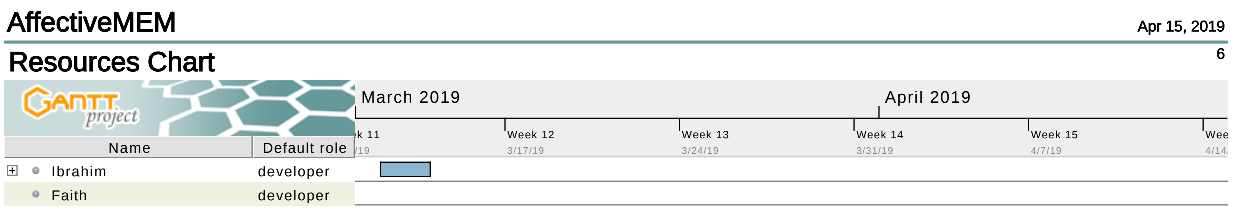
Name	Default role
Ibrahim	developer
Faith	developer



AffectiveMEM Apr 15, 2019

Gantt Chart 5

Name	E...	Begin date
Capture Video	...	3/12/19
Design	...	3/12/19
1.1.1 Class ...	...	3/12/19
1.1.3 Wirefr...	...	3/12/19
1.1.2 Expan...	...	3/14/19
Unit Test Capture	...	3/16/19
Unit Test Ca...	...	3/23/19
Unit Test CLI	...	3/23/19
Implement	...	3/16/19
Setup gi...	...	3/16/19
Install O...	...	3/18/19
Capture ...	...	3/18/19
CLI	...	3/18/19
Sample	...	3/26/19
Script to rec...	...	3/26/19
Collect data...	...	3/27/19
Script to loa...	...	3/28/19
Analyse Video	...	3/18/19
Sample	...	3/28/19
Collect data...	...	3/29/19
Script to loa...	...	3/28/19
Analysis	...	4/1/19
SSD	...	4/1/19
Design Class..	...	4/1/19
Wirefame	...	4/1/19
Implement	...	3/18/19
Setup FER L...	...	3/18/19
Setup Open...	...	3/18/19
Generate FE...	...	4/2/19
Generate Po...	...	4/3/19
Integrate B...	...	4/5/19
Integrate FER	...	4/5/19
Analyse Vid...	...	4/6/19
create com...	...	4/7/19
Evaluate	...	4/8/19
Unit Test	...	4/8/19
Code to Visu...	...	4/8/19
	...	4/8/19


AffectiveMEM

Apr 15, 2019

Resources Chart

6

Name	Default role
Ibrahim	developer
Faith	developer

Chapter 2: Appendix B

☐ Baseline Gantt Chart

☐ Tracking GanttChart

Part VI

Bibliography

Glossary

FER	Facial Expression Recognition.. 6

References

Deng, Z., Navarathna, R., Carr, P., Mandt, S., Yue, Y., Matthews, I., & Mori, G. (2017, Jul). Factorized variational autoencoders for modeling audience reactions to movies. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Retrieved from `http://dx.doi.org/10.1109/cvpr.2017.637` doi: 10.1109/cvpr.2017.637

Ratner, A., Hancock, B., Dunnmon, J., Goldman, R., & Ré, C. (2018). Snorkel metal: Weak supervision for multi-task learning. In *Proceedings of the second workshop on data management for end-to-end machine learning* (p. 3).

Saha, S., Navarathna, R., Helminger, L., & Weber, R. M. (2018). Unsupervised deep representations for learning audience facial behaviors. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 1132–1137).

Wu, T.-L., Wang, H.-K., Ho, C.-C., Lin, Y.-P., Hu, T.-T., Weng, M.-F., ... others (2008). Interactive content presentation based on expressed emotion and physiological feedback. In *Proceedings of the 16th acm international conference on multimedia* (pp. 1009–1010).

Zhang, S., Tian, Q., Jiang, S., Huang, Q., & Gao, W. (2008, Jun). Affective mtv analysis based on arousal and valence features. *2008 IEEE International Conference on Multimedia and Expo*. Retrieved from `http://dx.doi.org/10.1109/icme.2008.4607698` doi: 10.1109/icme.2008.4607698